AFRL-VS-TR-2002-1676

NASCAP-2K PRELIMINARY DOCUMENTATION

M.J. Mandell
V.A. Davis
I. G. Mikellides

Science Applications International Corporation
10260 Campus Point Drive
San Diego, CA 92121

22 May 2002

SCIENTIFIC REPORT NO. 2

20040303 214

**Air Force Research Laboratory
Space Vehicles Directorate
29 Randolph Road
Air Force Materiel Command
Hanscom AFB, MA 01731-3010**

This technical report has been reviewed and is approved for publication.

/Signed/
DAVID COOKE
Contract Manager

/Signed/
ROBERT MORRIS
Branch Chief

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE 05/22/2002	3. REPORT TYPE AND DATES COVERED Scientific Report No. 2

4. TITLE AND SUBTITLE

Nascap-2K Preliminary Documentation

6. AUTHORS
Myron J. Mandell
Victoria A. Davis
Ioannis G. Mikellides

5. FUNDING NUMBERS

C:F19628-98-C-0074
PE:63410F

PR:2822
TA:GC
WU:MX

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Science Applications International Corporation
10260 Campus Point Drive
San Diego, California 92121

8. PERFORMING ORGANIZATION REPORT NUMBER

SAIC 02/2021

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory
29 Randolph Road
Hanscom AFB, MA 01731-3010
Contract Manager: David L. Cooke/VSBXT

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-VS-TR-2002-1676

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This report is preliminary documentation for the three-dimensional Nascap-2k spacecraft-plasma environment interactions computer code. Nascap-2k is being developed as a successor to all the aforementioned codes as part of a program sponsored by the Air Force Research Laboratory and the NASA Space Environment Effects Program. Nascap-2k incorporates all the scientific modules of DynaPAC and uses the DynaPAC database. The following additional modules have been developed under the Nascap-2k program: (1) an Object Definition Toolkit (written in Java); (2) an analysis module (written in C++ and using the Boundary Element Method (BEM)2) for calculating surface charging in Geosynchronous Earth Orbit, Solar Wind or other tenuous plasma environments; (3) a graphical interface (written in Java) for setting up problems and examining results; and (4) a new GridTool program (Java) for building a DynaPAC multiply-nested grid structure.

14. SUBJECT TERMS			15. NUMBER OF PAGES 87
Spacecraft	Space environment	Plasma	
Nascap-2k	DynaPAC		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

TABLE OF CONTENTS

Table of Contents (Continued)

List of Figures

List of Figures (Continued)

List of Tables

1. OVERVIEW

1.1 Need for Nascap-2k

Both government and commercial satellite designers require advanced modeling capabilities to guide the design of satellites that can survive the natural environment. In the past, computer modeling of flight experiments (such as the SPEAR series) demonstrated excellent ability to predict steady-state interactions between high-voltage spacecraft and the ambient plasma. This ability was also extended to inherently dynamic problems involving three-dimensional space charge sheath formation, current flow in the quasi-neutral presheath, breakdown phenomena, plasma kinetics, ionization processes, and the effect of unsteady processes on spacecraft charging. *Nascap-2k*[1] takes advantage of improved understanding of all pertinent phenomena, as well as advances in computer technology, to make the improved modeling capability available to the spacecraft design community.

This document provides interim documentation of *Nascap-2k*. All of the existing relevant documentation was gathered into this one document and a few brief sections added to provide a preliminary source of information for users. Some aspects of the code are discussed in great detail and others totally ignored. Present plans call for developing more complete documentation as funding permits.

1.2 Background

The large number of existing and future spacecraft planned for launch in the near-term require three-dimensional computer codes that can predict spacecraft-plasma interactions under a variety of plasma conditions and orbits. Over the past two decades the fully three-dimensional computer codes *NASCAP/GEO*,[2] *NASCAP/LEO*,[3,4] *POLAR*,[4,5] and *DynaPAC*[6,7,8] have been developed to meet this need. While each code works well for the range of problems for which it was designed, by today's standards, these codes are very complicated to use. In addition *NASCAP/GEO* and *POLAR* are limited with respect to geometry.

Nascap-2k is being developed as a successor to all the aforementioned codes as part of a program sponsored by the Air Force Research Laboratory and the NASA Space Environment Effects Program. *Nascap-2k* incorporates all the scientific modules of *DynaPAC* and uses the *DynaPAC* database. The following additional modules have been developed under the *Nascap-2k* program: (1) an Object Definition Toolkit (written in Java); (2) an analysis module (written in C++ and using the Boundary Element Method (BEM)[2]) for calculating surface charging in Geosynchronous Earth Orbit, Solar Wind or other tenuous plasma environments; (3) a graphical interface (written in Java) for setting up problems and examining results; and (4) a new GridTool program (Java) for building a *DynaPAC* multiply-nested grid structure.

All information is stored in the database of the *DynaPAC* code[3], or as XML structured text files. *DynaPAC* features arbitrarily nested grids to provide good spatial resolution, and strictly continuous electric fields for accurate particle tracking. *Nascap-2k*'s geometric models and BEM charging results are accessible to *DynaPAC*'s modules, so that external potentials and particle trajectories can be calculated. We are using the *DynaPAC* synergy to extend the range of

applicability of *Nascap-2k* to denser plasma environments found, for example, in low-earth or polar orbits, or as a result of thruster plumes.

1.3 What Is Nascap-2k?

Nascap-2k is a graphical and interactive tool for studying plasma interactions with realistic spacecraft in three dimensions. Several screens are shown in Figure 1. *Nascap-2k* is targeted to spacecraft design engineers, spacecraft charging researchers, and aerospace engineering students. The simplified interface allows less experienced users to easily solve moderately complex plasma interactions problems. *Nascap-2k* also enables plasma interactions specialists to perform realistic analyses with direct application to engineering problems. Other than the charging module written specifically for *Nascap-2k*, the core physics capabilities are based on the *DynaPAC* modules. The core physics capabilities are as follows:

1) Solve for time-dependent potentials on spacecraft surfaces;

2) Solve the electrostatic potential about the object, with flexible boundary conditions on the object and with space charge computed either fully by particles, fully analytically, or in a hybrid manner;

3) Generate, track, and otherwise process representative macroparticles of various species in the computational space; and

4) View surface potentials, space potentials, macroparticle trajectories, and time-dependent potentials and currents.

The potential solver and particle tracking in *Nascap-2k* are centered around a DataBase Manager. The DataBase Manager is a library of routines capable of making large arrays of information contained in disk files accessible to computational modules. It has a programmer-friendly language for defining data types and for retrieving and storing data. The DataBase Manager is used to writing results in the format required for other application codes, and to retrieve information generated by other codes for use in *DynaPAC*. The DataBase strategy enables *Nascap-2k* to be operable on, and portable among, both personal computers using the Win32 platform and high-power UNIX workstations, which have proven to be more cost-effective than supercomputers for this type of code development and analysis.

Spacecraft geometrical definitions for *Nascap-2k* are done using Object Toolkit, which can import objects from other standard finite element pre-processors such as *PATRAN*. Among the advantages of this approach are that the geometry can be realistically represented and that finite element models of a spacecraft constructed for other purposes can be adapted for use in *Nascap-2k*. The computational space around the spacecraft is constructed interactively using the GridTool module. Arbitrarily nested subdivisions allow resolution of important object features while including a large amount of space around the spacecraft. The *Nascap-2k* module N2kDyn (also known as "Embed object in grid") constructs the properties of those finite elements neighboring the spacecraft.
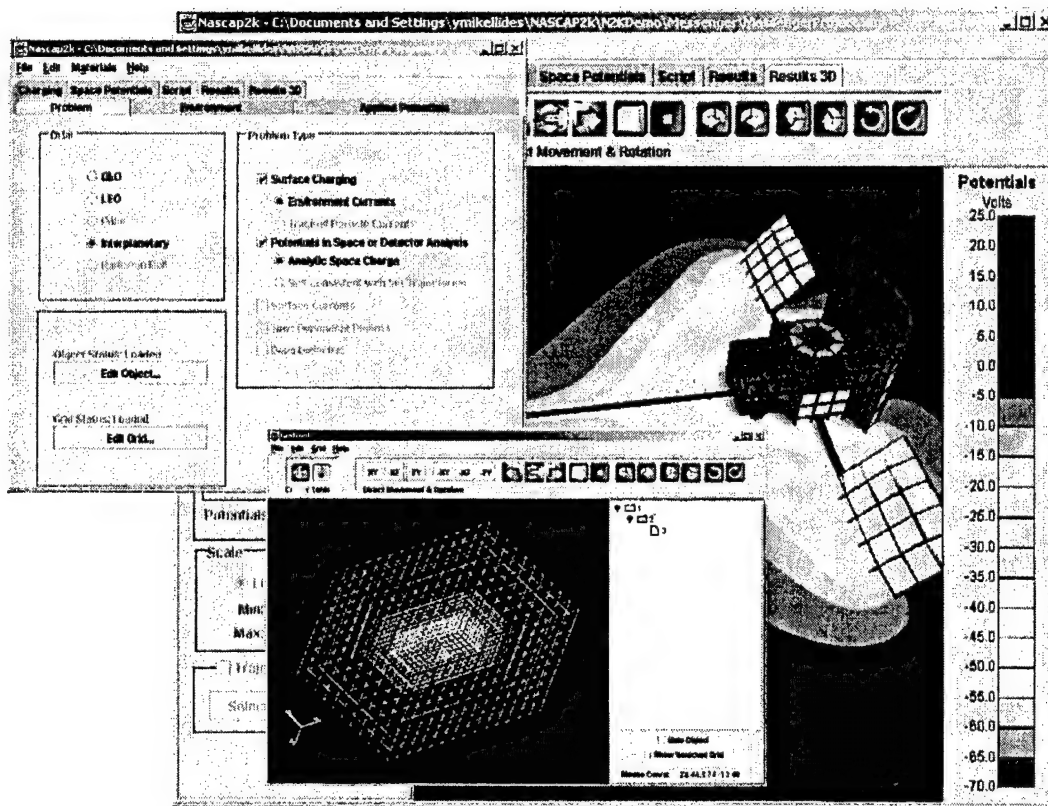
Figure 1. Selected Screens From *Nascap-2k* GUI.

Nascap-2k uses a high-order-finite element representation for the electrostatic potential that assures that electric fields are strictly continuous throughout space. The electrostatic potential solver uses a conjugate gradient technique to solve for the potentials and fields on the spacecraft surface and through the surrounding space. Space charge density models currently available include Laplacian, Linear, Non-linear, Frozen ions (immobile ions and barometric electrons, appropriate to the early stage of a negative transient pulse), Full Trajectory Ions, Hybrid PIC (appropriate to the several microsecond timescale response to a negative pulse), and Full PIC. The initial conditions input file for the potential solver is generated through the GUI.

Particle tracking is used to study sheath currents, to study particle trajectories, or to generate space charge evolution for dynamic calculations. *Nascap-2k* generates macroparticles either at a sheath boundary, a computational boundary, or throughout all space. Particles are tracked for a specified amount of time, with the timestep automatically subdivided at each step of each particle to maintain accuracy. The current to each surface cell of the spacecraft is recorded for further processing.

The "Results 3D" tab of the *Nascap-2k* GUI is used to generate graphical output illustrating such quantities as object surface potentials, space potentials, particle positions, and particle trajectories. Contour levels, plot range, *etc.* can be modified through the user interface.
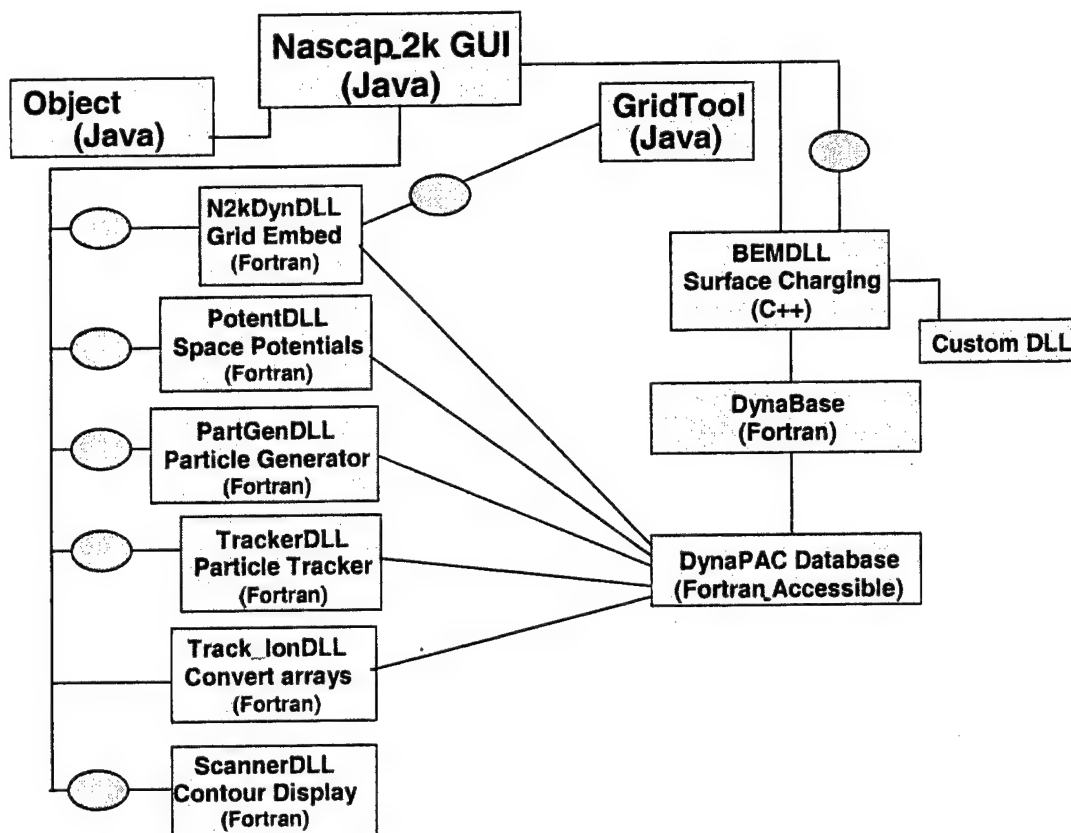
Figure 2. *Nascap-2k* Module Diagram. The solid fill rectangles represent *DynaPAC* modules converted to *Nascap-2k* DLLs, the line shaded rectangles represent software developed for *Nascap-2k*, and the ovals represent monitors.

The scientists and other researchers who contributed to this work include Dr. Myron. J. Mandell, Dr. Ira Katz, Mr. Jeffrey M. Hilton, Dr. Victoria A. Davis, Mr. David Monjo, Mr. Dale Lovell (summer intern), Ms. Barbara M. Gardner and Dr. Ioannis G. Mikellides.

2.   USING NASCAP-2K

2.1   Installation

Nascap-2k is installed by selecting Setup.exe on the installation disk. *Nascap-2k* requires the Java 1.4 runtime environment, including the Java3D extension.

Figure 3. Opening Screen of the *Nascap-2k* Tool.

2.2   Nascap-2k Structure

Nascap-2k consists of several modules as illustrated in Figure 2, with the bulk of the tool's scientific capability relying upon the *DynaPAC* (Dynamic Plasma Analysis Code) computer codes[6,7,8] for calculating potentials in three-dimensional space and on the BEM module for surface charging analyses. The suite of codes is written in Java (user interface), C++ and Fortran (science), and C (utility routines) and runs on the Win32 platform.

The modules communicate by XML files, keyword text input files, direct subroutine calls (DLL import/export), JNI subroutine calls, and the *DynaPAC* database. XML files and text files can be manually edited with a text editor or XML editor. XML Schemas specify the formats of the XML files. On Windows, the C++ modules validate their input files in accordance with the Schemas.

5

Object Toolkit is used to create finite-element representations of spacecraft surfaces. It also has materials editing capability, and can import *PATRAN* objects. It can also import objects from the *DynaPAC* database. Output (in XML) contains the recipe for recreating/reassembling the object, object definition by nodes and elements, and material definitions. Object Toolkit is described in Section 2.4.

GridTool is used to interactively define an arbitrarily nested cubic grid system for the space surrounding the object. GridTool is described in Section 2.5.

Nascap2K GUI is the user interface for *Nascap-2k*. It is based on an index-tab metaphor, and contains tabs for problem selection, initial conditions, parameter specification, script writing, time-dependent results analysis, and two- and three-dimensional display of surface potentials and fields.

BEMDLL (Charge surfaces) performs the Boundary Element Method analysis. It reads the object definition output file (XML), converts the object information to the *DynaPAC* structure, and stores it in the database. It exports standard methods and JNI methods.

DynaBase (Not user accessible) is the C++ callable gateway to the *DynaPAC* database.

Lapack (Not user accessible) is a custom implementation of matrix solver/inverter programs needed by *Nascap-2k*.

N2kDynDLL (Embed object in grid) reads the object definition and grid definition files and generates the surface and volume information required for potentials in space calculations.

PotentDLL (Potentials in space) is used to calculate the electrostatic potential field in the space surrounding the object.

PartGenDLL (Create particles) is used to generate macroparticles to study particle trajectories, surface currents, and space charge.

TrackerDLL (Track particles) is used to track macroparticles to study particle trajectories, surface currents, and space charge.

2.2.1   Units

The *Nascap-2k* user interface specifies the units of all input parameters. In general, *Nascap-2k* operates internally in the SI or MKS system of units. Electrostatic potentials are internally stored in volts, and electric fields in volts per meter. Magnetic fields are always in tesla (or webers per square meter). Particle energy and plasma temperature are in electron-volts. Charge density (ρ) is often divided by the permittivity of free space (ε_0), so that it has the units of volts per square meter.

2.2.2 Limits

Limiting values for the various problem elements are listed in Table 1.

Table 1. *Nascap-2k* Limits

Quantity	Limit
Grids	20
Points per grid	16383
Surfaces	4095
Nodes	4095
Special elements	8191
Conductors	20
Materials	20
Time Steps	1500
Particle types	20
Additional Points/Zone	100
Centroids/Zone	70
Triangles/Zone	150

2.2.3 Files

When initially opening a database, either with *Nascap-2k* GUI or GridTool a prefix is assigned to the file set. Table 2 describes the various files created by *Nascap-2k*.

Table 2. Files Created by *Nascap-2k*.

File	Contents
{*prefix*}.BS	Contains information about the bounding surfaces of the special elements, and is needed to calculate electric fields in special elements or to plot potentials in space.
{*prefix*}.DP	Main data file containing most of information about the calculation.
{*prefix*}.HI	Contains the database specifications, which characterize the data entities in the other files.
{*prefix*}.MEnn	Contain the potential solver matrices for special elements in grid nn. There is one such file for each grid containing special elements. They may be deleted if no additional calculations are anticipated.
{*prefix*}.POTG, {*prefix*}.POTS, {*prefix*}.CUR,	Files with stored histories of spatial potentials, surface potentials and surface currents respectively.
{*prefix*}.grd	ASCII file generated by GridTool that contains the grid information
Scratch.XX	Created by the potent module and may be deleted any time Potent is not operating. They should be deleted any time the potential calculation is reinitialized.
{*prefix*}_*_in.txt	Automatically generated ASCII input file containing well-labeled project parameters whose values may be altered by the user.
{*prefix*}_*_in.out	ASCII output file created by *Nascap-2k* module. The file contains a large amount of diagnostic information.
{*prefix*}Photo.xml	Contains the description of the photoemission spectrum.
{*prefix*}Object.xml	Contains the description of the the object.
{*prefix*}Project.xml	The project (xml) file. Contains all the information specified on the GUI. The best way to modify/update/run an existing problem is to copy this file, {*prefix*}.grd and "{*prefix*}Object.xml" into a new directory.
{*prefix*}BEM.BEM	Matrix elements generated and used by the BEM module.
ProjectDir.txt	Contains information used by the *Nascap-2k* GUI to determine the directory in which to look for files.

2.3 Problem Definition Tab

Figure 4 shows the problem definition screen of *Nascap-2k*. Shown are also the different problem tabs indicating the sequence of actions a user goes through when analyzing a spacecraft plasma environment interactions problem. This documentation follows the same sequence.
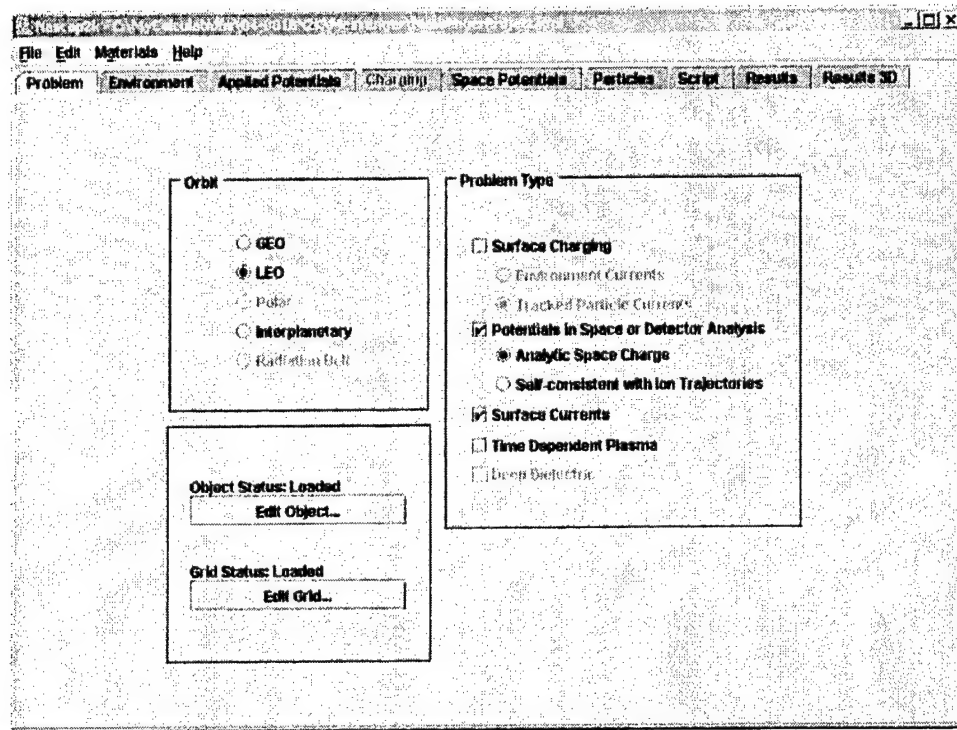
Figure 4. Problem Specification Tab in *Nascap-2k*.

2.4 Object Toolkit

The first step in the interactions assessment process between the environment and spacecraft is the construction of a geometrical model of the spacecraft.

Object Toolkit is used to create finite-element representations of spacecraft surfaces. It also has materials editing capability, and can import *PATRAN* objects. It can also import objects from the *DynaPAC* database. Output (in XML) contains the recipe for recreating/reassembling the object, object definition by nodes and elements, and material definitions. The Help Menu provides access to on-line documentation.

The screen for *Object Toolkit* looks as shown in Figure 5. The first four Cursor Tools are used to put the cursor in a mode to select specific objects, elements, edges, or nodes. The next two are used to put the cursor in a mode to translate or rotate an individual component. The final two Cursor Tools are used to put the cursor in a mode to translate or rotate the view. The Direct Movement and Rotation tools translate, scale, and rotate the view.

8
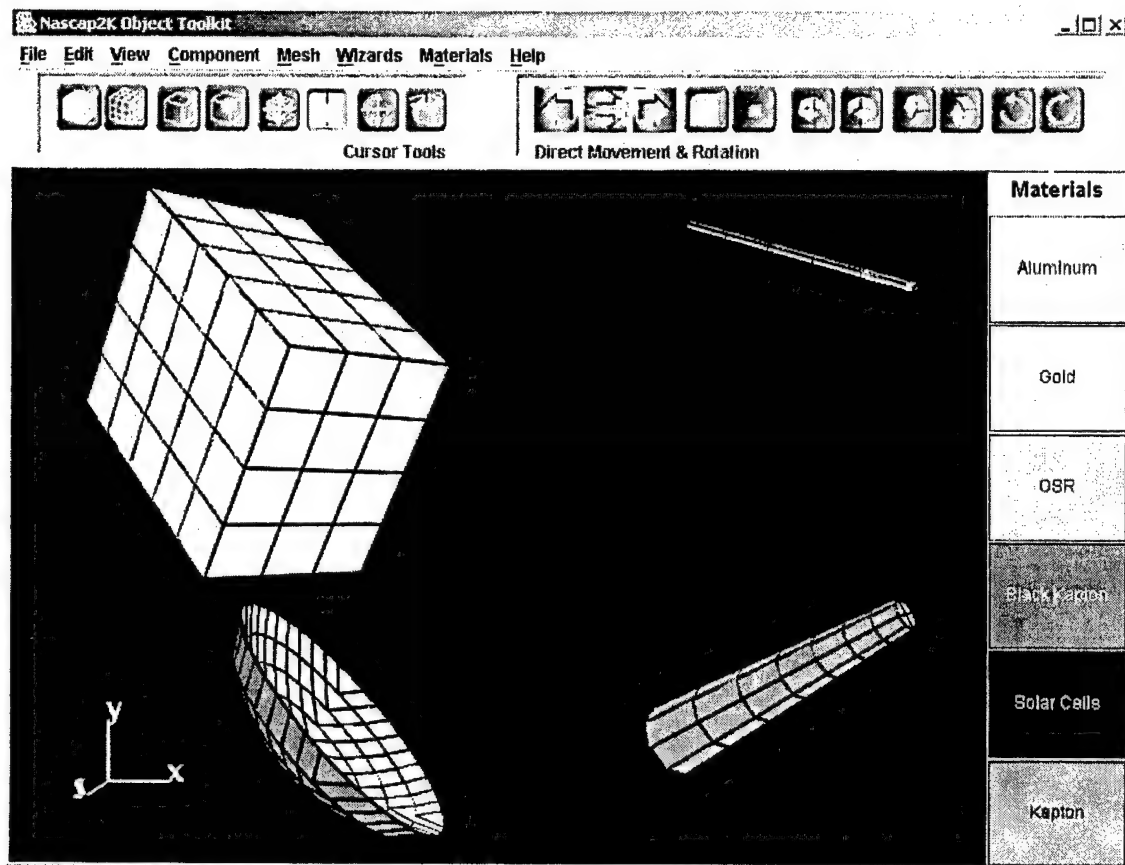
Figure 5. Object Toolkit Screen.

Object Toolkit builds an object as a hierarchical assembly structure. Each "Assembly" consists of two "Components." Each "Component" may be an "Assembly" or a primitive object. The available primitive objects are:

Box – A hexahedron (six faces, eight nodes, twelve edges), which is usually rectangular.

Boom – A subclass of Box with thin dimensions (one zone) in the X and Y directions.

Panel – A subclass of Box that is thin in the Z direction, and has the ±X and ±Y faces eliminated by sewing the edges together.

Dish – A primitive used to represent a parabolic antenna.

Cylinder – A primitive representing a 6-sided, 8-sided, or 4n-sided right cylinder (which may be tapered).

Primitive – An object defined by its mesh. Currently, a primitive can be created by converting a component, or by importing a *PATRAN* or *DynaPAC* object.

With each component of an assembly is associated a "Transformation" (from its local coordinates to the assembly's coordinates) describing how it is to be positioned within the assembly. Components can be crudely aligned and positioned using the mouse tools, and precisely aligned and positioned with some of the menu items described below. Certain types of attachments (or "welds") can be done (compatibly and without need for manual mesh editing) using "Wizards" provided as part of the tool.

9

The File menu includes " New," "Open," "Save," "Save As," "Save As BEM File," "Import *DynaPAC* Object." "Import SEE Handbook Materials File," and Exit.

The Edit menu includes "Undo," "Redo," "Cut," "Copy," "Paste," "Paste Special," "Edit Component," "Delete Component," and "Select All." The "Undo" and "Redo" are not fully implemented. Extreme caution must be exercised when Editing and Deleting components as elements may not match up in the same way after sizes of any components are changed. Mysteriously distorted elements and mysteriously changed materials on elements can be caused by elements no longer matching up as intended.

The View menu includes "Hide," "Hide Others," "Show All," "Material," "Conductor," and "Select View." Select View has options "From X Axis," "From Y Axis," "From Z Axis," "From -X Axis," "From -Y Axis," and "From -Z Axis."

The Components menu includes "Add New," "Add Component from File," "Node Relative Move," "Align Edge," "Center at Origin," "Enter Assembly," "Leave Assembly," "Top Assembly," "Consolidate Components into Assembly," and "Convert Component into Primitive." "Node Relative Move" is used to translate an assembly so that a node is at a specified location. "Align Edge" is used to rotate an assembly so that an edge is oriented as specified. These commands can be used to locate and orient the object for use by other tools, such as GridTool or *Nascap-2k* GUI. The three "Assembly" commands are used to select components that have been consolidated into Assemblies. Once a component has been converted to a primitive, only mesh editing is available.

The Mesh menu includes "Materials," "Conductors," "Create Element," "Delete Element," "Divide Element," "Combine Triangles," "Reverse Elements," "Edit Node," "Combine Nodes," and "Rebuild Mesh." "Materials" and "Conductors" is used to change the material or conductor on the selected element. "Rebuild Mesh" does a comprehensive rebuilding of the mesh, including renumbering. The mesh is usually only partially rebuilt each time a change in made in the object. With experience, we hope to determine all those cases where additional rebuilding is necessary and automate it, making this menu choice unnecessary.

There are five wizards that move components so that they join. Components combined using a wizard are automatically joined into assemblies. The five wizards are "Element to Element," "Edge to Element," "Element to Edge," "Edge to Edge," and "Surface to Surface." After using the "Surface to Surface" wizard it is usually necessary to do some mesh editing as joining two surfaces is too complex for easy automation.

The materials menu permits creating, deleting, and editing materials. Editing of which material is on a specific surface is set when defining the component and by editing a specific surface using the Materials choice on the Mesh menu.

Object Toolkit can import objects created using *PATRAN*. The *PATRAN* file must have a filename ending in "pat." Material and conductor number definitions are not preserved. The *PATRAN* object must be constructed of linear triangles (TRI/3/icond) and linear quadrilaterals (QUAD/4/icond).

Object Toolkit can also read in a previously defined object from a *DynaPAC* database. A user can use the *DynaPAC*[5] codes PatDyn, MenDyn, and NasDyn to read in a *PATRAN*, *Mentat*, or *NASCAP/GEO* object into a *DynaPAC* database and then read the object back out into Object Toolkit. The object can then be written out as an xml file for use in *Nascap-2k*.

2.5   GridTool

In *Nascap-2k* the GridTool module is used to define an arbitrarily nested grid structure about the object. GridTool allows the user to define a grid structure for an object created by Object Toolkit or contained in a *DynaPAC* database. It is a Java application.

GridTool presently has the following capabilities:

1) Create and modify a primary grid around an object.
2) Add and modify a child grid.
3) Delete a grid along with all of its child grids.
4) Restart from an existing grid structure.
5) Graphically display the current grid structure.

To generate and modify a grid in space through the *Nascap-2k* GUI, the user must chose "Edit Grid" on the *Nascap-2k* problem specification tab (it is assumed that an object has already been defined). To better illustrate the process a simple example of a 3-level grid for a two-cubes problem is shown in Figure 6.

11
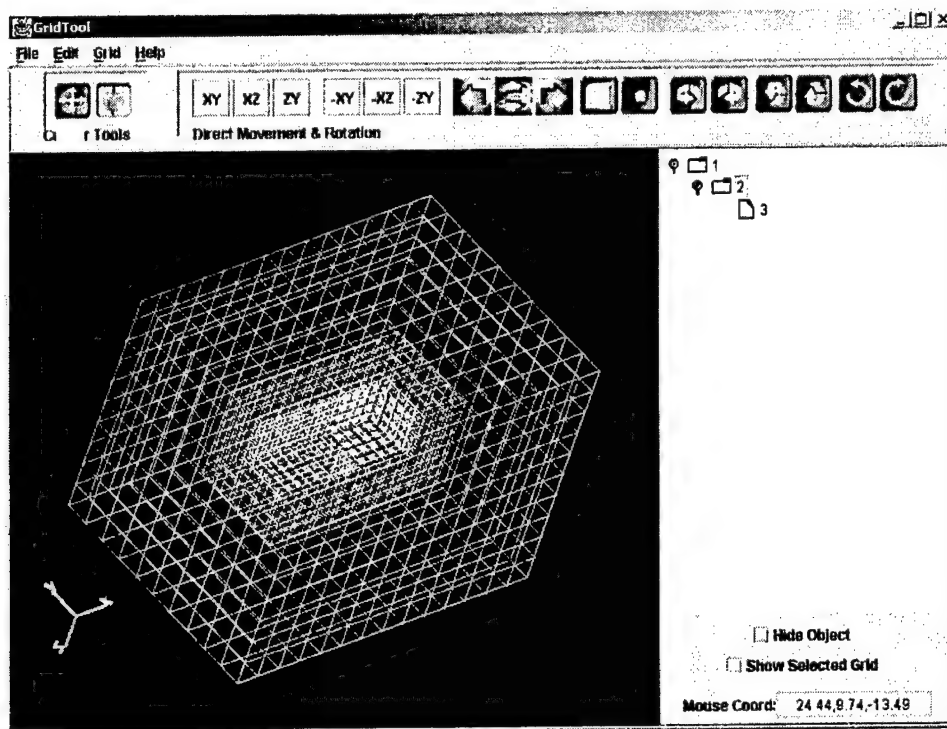
Figure 6. Grid in Space for a Two-cubes Problem Generated Using GridTool.

The user can choose the parent grid or any of the child grids to modify or delete, and/or add a new child grid. In the example above the second-level child grid has been chosen for modification. The options are shown in Figure 7.

Figure 7. Sample (GridTool) Window in Nascap-2k Used for Modifying Grids in Space.

12

Grid files are saved with the extension ."grd." This file is identical to the *DynaPAC* "grd" file. The contents of the grid file for the TwoCubes problem (see Figure 6) are shown in Table 3.

Table 3. Interpretation of the Contents in *prefix*.grd

Content in	Line	Field(s)	Defines
3	1	1	Total number of grid levels
1 15 11 11 0 1 15 1 11 1 11	2	1	Grid level
		2-4	Nx, Ny, Nz
		5	PARENT grid level number
		6-11	Limits in PARENT grid coordinates
1.0 15.0 1.0 11.0 1.0 11.0 1.1	3	1-6	Limits in PRIMARY grid coordinates
		7	Mesh size (meters)
1 1	4	1	Mesh ratio wrt PARENT grid
		2	Mesh ratio wrt PRIMARY grid
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	5	all	Future use
2 17 9 9 1 4 12 4 8 4 8	6	Same as line 2	Same as line 2
4.0 12.0 4.0 8.0 4.0 8.0 0.55	7	Same as line 3	Same as line 3
2 2	8	Same as line 4	Same as line 4
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	9	Same as line 5	Same as line 5
3 25 9 9 2 3 15 3 7 3 7	10	Same as line 2	
5.0 11.0 5.0 7.0 5.0 7.0 0.275	11	Same as line 3	Same as line 2
2 4	12	Same as line 4	Same as line 4
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	13	Same as line 5	Same as line 5
6.0 2.0 2.0 0.0 0.0 0.0 1.0	14	1-3	Object size (meters)
		4-6	Object center relative to PRIMARY grid center
		7	Unit conversion factor

2.6 Specifying the Environment

Presently, *Nascap-2k* supports geosynchronous, low-earth orbit, and interplanetary environments. The environmental parameters are specified on the Environment tab. Common to all three are the specification of the local magnetic field vector and the direction from the spacecraft to the Sun and (relative) intensity of the Sun at the location of the spacecraft.

2.6.1 GEO Environment Tab

The tab for definition of the geosynchronous environment is shown in Figure 8. For the geosynchronous environment, three predefined environments are available. The predefined environments include the NASA Worst case, ATS-6, and the Sept 4, 1997 environments. (The Sept 4, 1997 environment is a fit to a real environment measured by one of the LANL MPA instruments that was used once to model the behavior of another spacecraft with minimal to moderate success. The values for the predefined environments are summarized in Table 4. Otherwise, the user may select "User Defined" and specify the electron and ion density and temperature. Either a single Maxwellian or a double Maxwellian environment can be specified.

Figure 8. The Environment Screen for Studies in Geosynchronous Orbit Environment.

Table 4. Available Predefined Geosynchronous Environments

	Worst Case	ATS-6	Sept. 4th 1997
Electron Density (m⁻³)	1.1e6	1.2e6	3e5 – 2e5
Electron Temperature (eV)	1.2e4	1.6e4 – 1000	4000 - 7000
Ion Density (m⁻³)	2.4e5	2.4e5 – 8820	3e5 – 2e5
Ion Temperature (eV)	2.95e4	2.95e4 – 111	4000 - 7000
Electron Current (A/m²)	3.3e-6	4.1e-6	9.6e-7
Ion Current (A/m²)	2.5e-8	2.5e-8	2.2e-8
Distribution	Single Maxwellian	Double Maxwellian	Double Maxwellian

2.6.2   LEO Environment Tab

The tab for definition of the low-earth-orbit environment is shown in Figure 9. The plasma density and temperature are specified. The Debye length is computed from the density and temperature. If the user changes the Debye length the density is recomputed. The spacecraft's velocity vector is specified here. The particle species are defined here.

Figure 9. The Environment Screen for Studies in LEO.

2.6.3   Interplanetary Environment Tab

The tab for definition of interplanetary environment is shown in Figure 10. The densities of the ion and electron species are assumed to be the same. The electron temperature and ion velocity are specified. The ions are assumed to be hydrogen streaming from the same direction as the Sun. The ion velocity can also be specified by the ion energy, in which case the velocity is automatically computed from the energy.

Figure 10. The Environment Screen for Studies in the Interplanetary Space.

2.6.4 Material Property Editing

The properties of any material can be changed using the edit material properties screen shown in Figure 11. The screen is accessed by selecting the material name from the Material menu. We recommend the *SEE Spacecraft Charging Handbook* as an aid to the determination and preliminary assessment of material property sets.

Figure 11. Material Properties Screen.

2.6.5 Using Photoemission Spectra

Spacecraft in the Solar Wind normally charge to positive potentials (a few volts to tens of volts) to balance emitted photoelectron current with a very low current of ambient electrons. Determining just how positive a spacecraft will charge requires more detailed knowledge of the photoelectron spectrum than is required for geosynchronous charging, in which potentials are negative and of much larger magnitude. Pushing the Edit Photoemissions button on the Material property definition screen accesses the photoemission spectrum. A *Nascap-2k* definition of a photoemission spectrum consists of the total emission and a sequence of energy-value pairs, where the value indicates the fraction of the photoelectron spectrum lying above the energy. If no spectrum is specified, the photoelectron spectrum is assumed to be a 2 eV Maxwellian.

Total Emissions (Am⁻²): 2.000E-5

Energy (eV)	Fraction of spectrum above
0.0	1.000
5.180	0.108
5.930	8.760E-2
7.260	6.410E-2
9.410	4.170E-2
12.80	2.330E-2
18.30	1.190E-2
27.25	5.220E-3
41.77	1.540E-3
65.24	2.400E-4

Figure 12. Photoemission Screen.

2.7 Applied Potentials Tab

Figure 13 shows the Applied Potentials tab. This tab is used to specify applied and initial potentials on conductors and insulating surfaces. The top portion is used to specify if each conductor is held at a fixed potential, allowed to float, or held at a fixed bias with respect to another conductor. Potentials on conducting materials are generally specified by specifying the potential on the underlying conductor. The lower part of this tab is used to specify the initial potential on surface elements. Surfaces are selected by material name, conductor number, and sunlit or dark condition. The specific surface number can also be specified. Presently, only fixed potential boundary conditions can be specified here.

Figure 13. Potential Initialization for Objects in the Two-cubes Sample Problem.

2.8 Surface Charging Using the BEM Module

The Charging tab shown in Figure 14 is used to specify the timestepping parameters used in calculations of surface charging by the BEM module.

Figure 14. Tab for Specifying Parameters Used in Spacecraft Charging Calculations Using the BEM Module.

2.8.1 Physics and Numeric Background

Spacecraft surface charging is the accumulation of charge on spacecraft surfaces. The surfaces of geosynchronous spacecraft can accumulate charge due to incidence of energetic (10 to 50 keV) substorm electrons. As illustrated in Figure 15, several different currents contribute to the charging.[9,10,11,12] Kilovolt electrons generate secondary electrons and can be backscattered (reflected from surfaces). Kilovolt ions can also generate secondary electrons. The current density of low-energy electrons generated by solar ultraviolet emission exceeds that of the

18

natural charging currents. The rest of the spacecraft influences the potential of each surface. In order to compute surface potentials, spacecraft geometry, surface materials, and environment must all be considered. Each insulating spacecraft surface interacts separately with the plasma and is capacitively and resistively coupled to the frame and other surfaces. *Nascap-2k* uses this information to compute the time history of the surface potentials and fluxes.
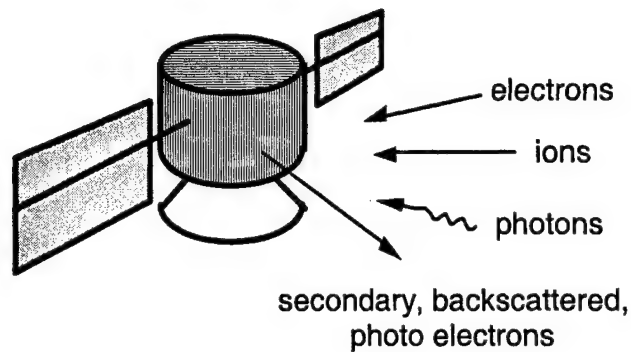
Figure 15. High Negative Potentials Can Result From the Accumulation of Charge on Spacecraft Surfaces.

Figure 16 shows a circuit diagram for a spacecraft with one insulating surface and exposed conducting surfaces. The widely differing capacitances of the surface to infinity, C_A and the capacitance of the surface to spacecraft ground, C_{AS}, make this a complex numeric problem.

$$C_{AS} = \kappa \varepsilon_0 \frac{S}{d} \approx \frac{S}{2} \times 10^{-7} \text{ Farad}$$

$$C_A \approx C_S \approx 4 \pi \varepsilon_0 \, r \approx r \times 10^{-10} \text{ Farad}$$
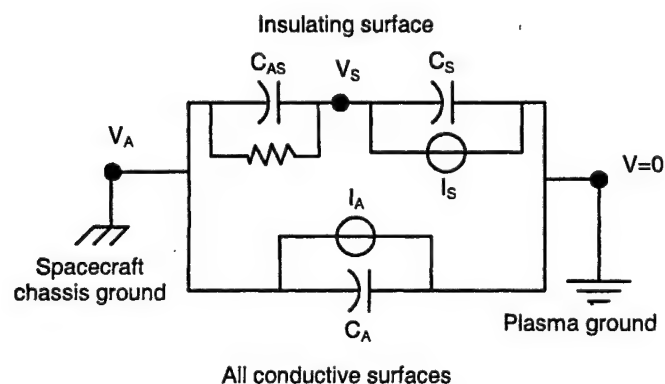
Figure 16. Circuit Model of a Spacecraft With One Insulating Surface.

The potentials as a function of time are computed using implicit time integration of the charging equations.

$$C_A \dot{V}_A + C_{AS} (\dot{V}_A - \dot{V}_S) = I_A \tag{1}$$
$$-C_{AS} (\dot{V}_A - \dot{V}_S) + C_S \dot{V}_S = I_S$$

The multisurface problem is solved by linearizing the currents and inverting the matrix.

$$\mathbf{C} \dot{V} = \mathbf{I}(V) \tag{2}$$

2.8.1.1 Boundary Element Method Algorithm

Among the difficulties of developing accurate and robust algorithms for spacecraft charging has been the inability to calculate electric fields accurately, let alone to predict how electric fields will change as a result of surface potential changes. We proposed at the AFRL Spacecraft Charging Conference (November 1998) and at the AIAA Aerospace Sciences Meeting (January 1999) using the Boundary Element Method to calculate accurate electric fields, and as the basis for implicit charging equations.

The Boundary Element Method (BEM)[13] is a means for relating fields and potentials in a region to sources on the boundary of the region. It is comparable to a sum over the coulomb field of all the charges in a region rather than an iterative field solution. In our case, the region is the space exterior to a spacecraft, and the boundary is the spacecraft surface. Also, we assume the "free space Green's function," *i.e.,* the potentials in the region will obey Laplace's equation.

The sources are sheets of charge coincident with the spacecraft model's surface elements. We assume that each surface element, j, has a constant charge density, σ_j. The familiar relation for the potential of a point charge then generalizes to an integral over the object surface:

$$V = \frac{q}{4\pi\varepsilon_0 r} \rightarrow (4\pi\varepsilon_0)V_i = \sum_j \int d^2 r_j \frac{\sigma_j}{r_{ij}} \tag{3}$$

where V_i, which could be the potential at any point in space, is considered the potential at the center of a surface element. Similarly, the familiar relation for the electric field of a point charge generalizes to:

$$E = \frac{q}{4\pi\varepsilon_0 r^2} \rightarrow (4\pi\varepsilon_0)\mathbf{E}_i = \sum_j \int d^2 r_j \frac{\sigma_j}{r_{ij}^3}(\mathbf{r}_i - \mathbf{r}_j) \tag{4}$$

where E_i is the electric field at some point in space, the point again taken at the center of a surface element.

We express the relations of potential and electric field to charge density as matrices:

$$V_i = [\mathbf{C}^{-1}]_{ij}\sigma_j \qquad\qquad \mathbf{E}_i \cdot \mathbf{n}_i = \mathbf{F}_{ij}\sigma_j \tag{5}$$

These can be combined to obtain a relation between normal electric field and voltage:

$$\mathbf{E}_i \cdot \mathbf{n}_i = \mathbf{F}_{ik}\mathbf{C}_{kj}V_j \tag{6}$$

This last relation is the key to developing relations between surface charge, surface potential, and surface currents in order to derive charging equations.

2.8.1.2   Doing the Integrals

To get C^{-1} we need to do integrals of the form

$$\int d^2 r_j \frac{\sigma_j}{r_{ij}} \tag{7}$$

There are tricks to doing these integrals, which can be found in the literature. Denote the field point, r_i as P, and take the domain of integration as triangle ABC. Then, the vector from the field point to anywhere on the triangle can be parameterized as

$$\mathbf{r}_{ij} = PA + uAB + uvBC \tag{8}$$

where PA, AB, and BC are vectors between pairs of points, and u and v are parameters, each in the interval [0,1]. The square of the distance, r_{ij}, then becomes

$$r_{ij}^{\,2} \equiv \mathbf{r}_{ij} \cdot \mathbf{r}_{ij} = \sum_{0 \le k,l \le 2} T_{kl}u^k v^l \tag{9}$$

where the coefficients T_{kl} are pairwise scalar products of the three vectors above. Now, the integral can be expressed as

$$\int d^2 r_j \frac{\sigma_j}{r_{ij}} = \int_0^1 dv \int_0^1 udu \frac{\sigma_j}{\sqrt{V_0 + V_1 u + V_2 u^2}} \tag{10}$$

where the V_i coefficients are functions of v.

The inner integral (over u) may be found in standard integral tables[14]:

$$\int x dx \frac{1}{\sqrt{c+bx+ax^2}} = \frac{\sqrt{c+bx+ax^2}}{a} - \frac{b}{2a}\int dx \frac{1}{\sqrt{c+bx+ax^2}}$$

$$\int dx \frac{1}{\sqrt{c+bx+ax^2}} = \frac{1}{\sqrt{a}}\log\left|2\sqrt{a}\sqrt{c+bx+ax^2}+2ax+b\right| \tag{11}$$

21

We are left to do the outer integral (over v) numerically. To facilitate this, we select the vertex A such that the scalar product PA·BC is the minimum of the three choices. Then, very few integration points are needed in the outer integral (over v). (We use a 5-point Simpson's rule in the current implementation.)

The integrals for the electric field are similar, although there are more of them. The same techniques apply.

2.8.1.3   Charging Equations

To develop charging equations we need to express physical charges on physical surfaces in terms of the voltages on the same objects. We can then proceed to calculate what voltage changes will be produced by changes in charge (currents). Because the interior of a spacecraft is NOT free space, the physical charge densities bear no relation to the σ's, which we have now eliminated from the calculation.

To get the physical charges we use Maxwell's divergence equation in the form $\sigma = \nabla \cdot \mathbf{D}$. Figure 17 shows the "Gaussian pillboxes" used to calculate the actual surface charges on insulating surfaces and on conductors.
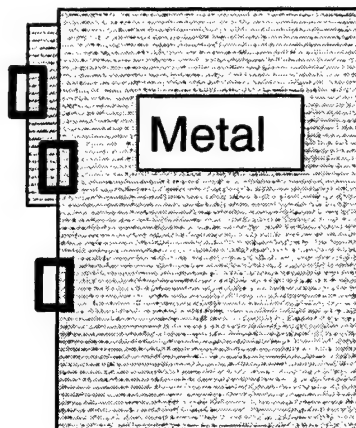
Figure 17. "Gaussian Pillboxes" Used to Calculate the Actual Surface Charges on Insulating Surfaces and on Conductors.

For an insulating surface, the external field is $\mathbf{E} \cdot \mathbf{n}$, which we obtain from the BEM solution. The internal field is related to the capacitance between the insulating surface and its underlying conductor. Thus, the total charge, q_i on such a surface is given by:

$$q_i = A_i (\mathbf{E} \cdot \mathbf{n})_i + C_{ic} (V_i - V_c) \tag{12}$$

For a conductor, since charges are mobile, it is not useful to know the charge on each individual surface, but we need to work with the total charge on the conductor. Conducting surfaces include the obvious "bare" surfaces, as well as the surfaces that underlie the insulating surfaces. In both cases, we have zero electric field internal to the metal. To obtain the total charge on all the

22

surfaces of the conductor, we need to sum the external-field charge terms for the bare conducting surfaces, plus the capacitive-charge terms for the insulator-metal interfaces:

$$Q_c = \sum_{bare} A_i (\boldsymbol{E} \cdot \boldsymbol{n})_i - \sum_{insulators} C_{ic} (V_i - V_c) \tag{13}$$

We previously found the BEM expression for the external fields in terms of the cell potentials:

$$\boldsymbol{E}_i \cdot \boldsymbol{n}_i = \boldsymbol{F}_{ik} \boldsymbol{C}_{kj} V_j \tag{14}$$

Substituting this into the equations for q_i and Q_c, performing the indicated sums, and adding the capacitive terms, we get the matrix equation:

$$\boldsymbol{Q} = \boldsymbol{G} \boldsymbol{V} \tag{15}$$

where the vectors are composed of contributions from all the insulating surfaces and a single term representing all the bare conducting surfaces.

$$\boldsymbol{Q} = \{ q_1, q_2, \dots, q_n, Q_c \} \tag{16}$$
$$\boldsymbol{V} = \{ V_1, V_2, \dots, V_n, V_c \}$$

where $\boldsymbol{Q}$ and $\boldsymbol{V}$ only contain entries for those entities physically capable of accumulating charge, *viz.*, conductors and insulating surfaces, and the charges and potentials are related by the charging matrix, $\boldsymbol{G}$.

We are looking for an equation that relates currents to voltage changes. So, we differentiate the charge equation in time:

$$\boldsymbol{I} \equiv \dot{\boldsymbol{Q}} = \boldsymbol{G} \dot{\boldsymbol{V}} \tag{17}$$

Discretize to a finite time interval:

$$\boldsymbol{I} \Delta t = \boldsymbol{G} [\boldsymbol{V}(t + \Delta t) - \boldsymbol{V}(t)] \tag{18}$$

Implicitize by evaluating current at the final time (also, for simplicity, changing [t+Δt,t] to [t,0]):

$$\boldsymbol{I}(t) t = \boldsymbol{G} [\boldsymbol{V}(t) - \boldsymbol{V}(0)] \tag{19}$$

Linearize currents with respect to voltage (since we do not know the final voltages at which the current is to be evaluated):

$$\boldsymbol{I}(t) \approx \boldsymbol{I}(0) + \boldsymbol{I}' [\boldsymbol{V}(t) - \boldsymbol{V}(0)] \tag{20}$$

And, solve the equation:

$$[\boldsymbol{G} - \boldsymbol{I}' t] \boldsymbol{V}(t) = [\boldsymbol{G} - \boldsymbol{I}' t] \boldsymbol{V}(0) + \boldsymbol{I}(0) t \tag{21}$$

23

This gives us a straightforward matrix equation. Everything on the right-hand-side is known, and we can solve using standard linear algebra equation solver packages.

Before proceeding to examples, it is worth commenting on the derivative of current, $\mathbf{I'}_{ij} = \partial I_i / \partial V_j$. Consider three cases:

For current sources such as incident plasma current, we usually approximate the current as a function of the local voltage only. This gives a diagonal term in $\mathbf{I'}_{ij}$. Since such currents decay exponentially, some care must be taken not to underestimate the final current if the voltage is changing in such a direction that the current is increasing (i.e., electron current for a surface whose potential is increasing (toward zero) from a large negative potential).

Conductivity current, such as $I_i = \sigma(V_i - V_c)$, contributes off-diagonal as well as diagonal terms to the current derivative matrix. Surface conductivity contributes many more off-diagonal terms.

The case that causes great difficulties for *NASCAP/GEO* is $I_i = I_i(E \cdot n)$. This occurs for a photo-emitting surface at negative potentials. The problem is that the local electric field is a function of the potentials on all the surfaces. But, the BEM provides us with exactly that function. So, we can now compute the term,

$$I'_{ij} \equiv \partial I_i / \partial V_j = \partial I_i / \partial E_i \times \partial E_i / \partial V_j \tag{22}$$

using the relation

$$\mathbf{E}_i \cdot \mathbf{n}_i = \mathbf{F}_{ik} \mathbf{C}_{kj} V_j \tag{23}$$

derived from the BEM.

2.8.1.4   Example: Sunlit Sphere

We are now ready to recalculate the charging of a sunlit Teflon sphere in a 1 cm^{-3}, 20 keV plasma. The *NASCAP/GEO* version of this was published in 1978.[15] A modern version of the original result is shown below in Figure 18. The dark side of the sphere gradually charges negative due to incident plasma electrons, while photoemission grounds the sunlit side. The Sun direction is (1,1,1) (from the upper right). Eventually, however, the strong negative potentials due to the dark surfaces "wrap around" the sphere and form a barrier to photoelectron escape. The potential of each sunlit surface is subsequently determined by the condition that its electron field has a small, positive value (too small to be seen in Figure 18), so that just the right fraction of photoelectrons can escape over the barrier.

24
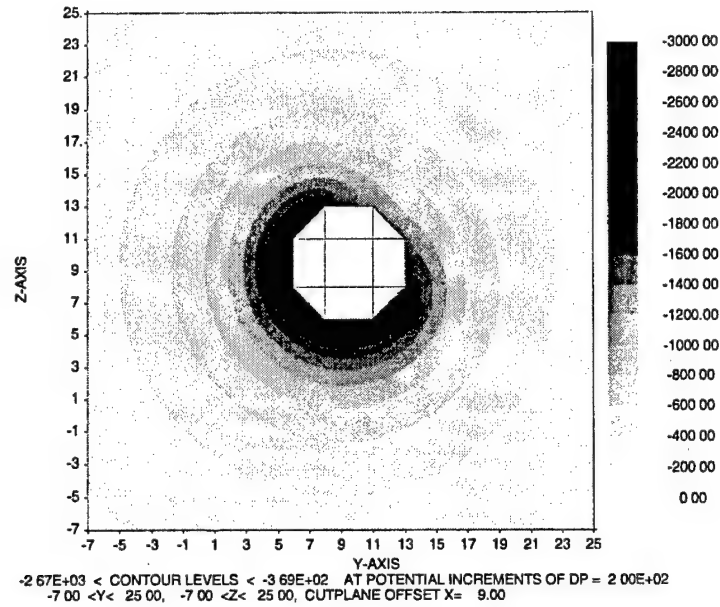
Figure 18. Potentials About Sunlit QSphere in Space as Computed by *NASCAP/GEO*.

Figures 19 and 20 show the BEM solution (using *DynaPAC* graphics). The subsolar point is the least negative point.
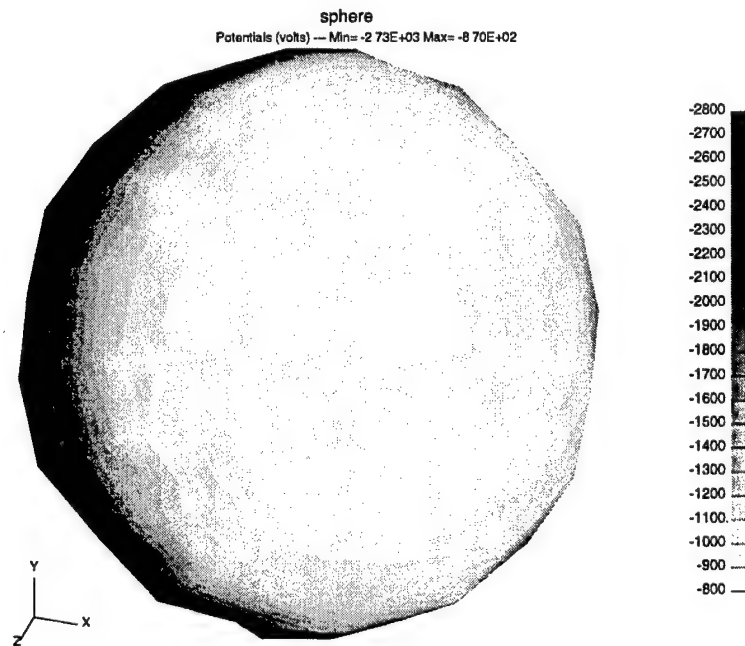
Figure 19. Potentials on Sunlit *PATRAN* Sphere in Space Viewed From Direction (1,2,3) as Computed by BEM.

Qsphere
Potentials (volts) — Min= -2 60E+03 Max= -8 47E+02
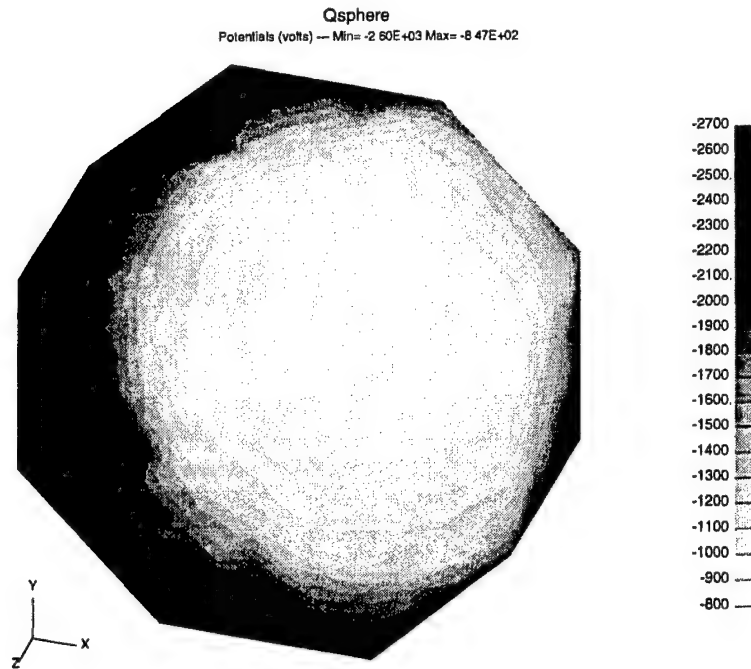
Figure 20. Potentials on Sunlit QSphere in Space Viewed From Direction (1,2,3) as Computed
by BEM.

Figure 21 is another view of the BEM solution, showing more of the dark side. Despite the fairly
coarse gridding on the sphere, the gradual potential gradient on the sunlit side and the constant,
large negative potential on the dark side are clearly seen in the BEM solution.

sphere
Potentials (volts) — Min= -2 73E+03 Max= -8 70E+02
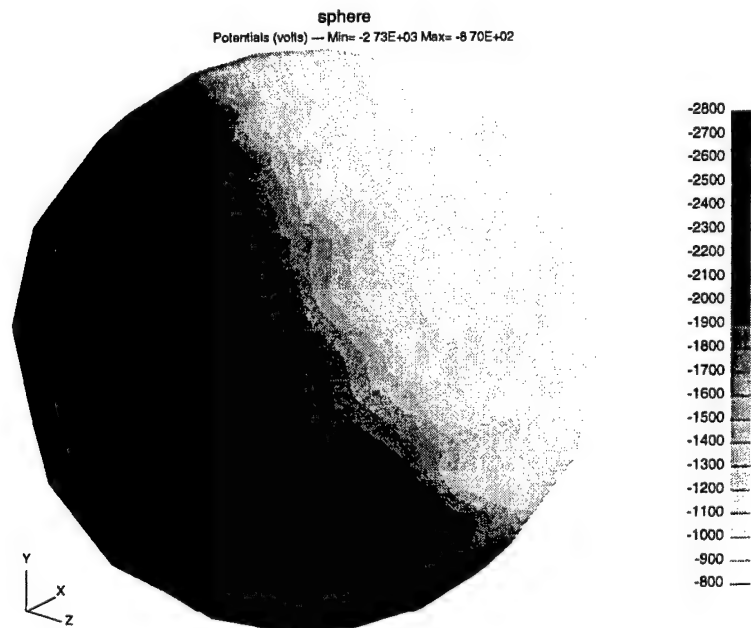
Figure 21. Potentials on Sunlit *PATRAN* Sphere in Space as Computed by BEM.

26

The table below compares the original *NASCAP/GEO* solution with the current *NASCAP/GEO* and *Nascap-2k* BEM solutions.

Table 5. Results Comparison.

	NASCAP/GEO 1978	NASCAP/GEO 1999	BEM QSphere	BEM Sphere
Min. Potential	-3600	-2700	-2880	-2730
Max. Potential	-1000	-1000	-1160	-870
Min. Field			-18300	-5550
Max. Field			4.64	4.58

Note that, even though the same input is used, we can no longer reproduce the original *NASCAP/GEO* 1978 result because the treatments of secondary emission, backscatter, etc. have changed in unknown ways. But, the general character of the solution is the same, and differences among the remaining calculations are all plausible. In particular, the difference in minimum electric field (which occurs in the dark area near the terminator) is attributable to the smaller cell size for the QSphere compared with the *PATRAN* sphere. (Since we currently have no way of defining a QSphere for *DynaPAC* on the Win32 platform, the QSphere was manually input to the BEM module, and no graphics were available.) We did not keep careful track of the timestepping, but the case of all negative potentials (with photo-emission currents depending on electric field) seemed to go very smoothly.

2.9   Space Potentials Tab

Figure 22 illustrates the space potentials tab. This tab is used to specify the parameters used in potentials in space calculations (besides the environment) that users are most likely to wish to change. Additional parameters are provided on the Advanced tab.
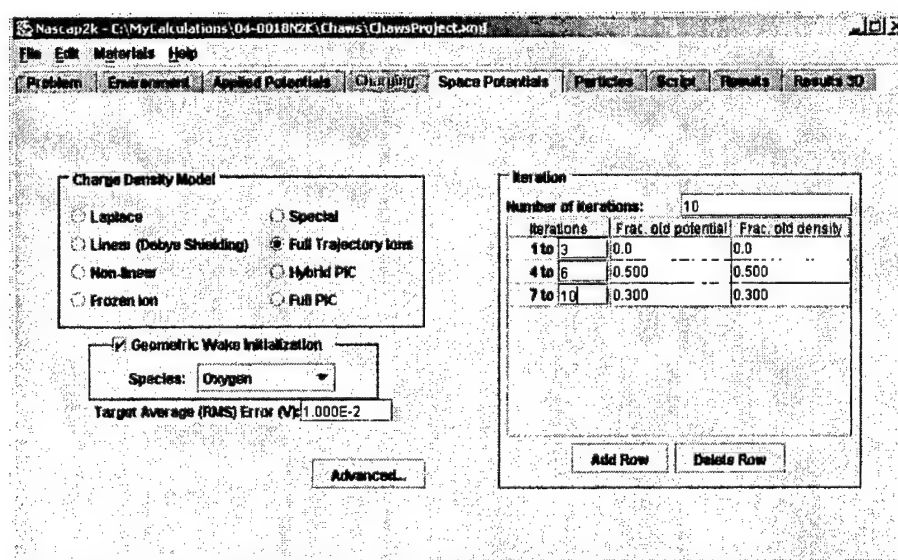
Figure 22. Space Potentials Tab.

2.9.1 Description of Menu Parameters

The potentials in space calculation solves Poisson's equation with a user specified charge density model. The available charge density models are described below.

Laplace
The Laplacian space charge option specifies that the charge density is zero.

$$-\nabla^2\phi = 0 \tag{24}$$

i.e., charge exists only on object surfaces and external boundaries, as determined by the boundary conditions. "Space charge" iterations may still be required, however, due to the treatment of surface electric fields.

Linear (Debye Shielding)
The Linear space charge option solves the Helmholtz or Debye-Huckel equation

$$-\nabla^2\phi = -\phi/\lambda^2 \tag{25}$$

The value of λ is the maximum of the plasma Debye length (divided by the square root of density depletion due to geometric shadowing), and the local mesh spacing divided by the DEBLIM parameter (which defaults to 2).

Non-Linear
Non-linear space charge is calculated using the *NASCAP/LEO* formulation:

$$-\nabla^2\phi = -\rho/\varepsilon_0$$

$$\rho/\varepsilon_0 = -\left(\phi/\lambda_D^2\right)\frac{1+|\phi/\theta|C(\phi,E)}{1+\sqrt{4\pi}|\phi/\theta|^{3/2}} \tag{26}$$

$$C(\phi,E) = |\theta/\phi|\left[(R_{sh}/r)^2 - 1\right]$$

$$(R_{sh}/r)^2 = 2.29|E\lambda_D/\theta|^{1.262}|\theta/\phi|^{0.509}$$

Where,
ρ = space charge (C/m)
ε_0 = permittivity of vacuum (8.854e-12 F/m)
λ_D = plasma Debye length (m)
θ = plasma temperature (eV)
ϕ = local space potential (V)
E = local space electric field (V/m)

The last equation (analytic focusing) comes from fitting a finite temperature spherical (Langmuir-Blodgett) sheath. If analytic convergence is turned off (by the CONV input parameter) then $C(\phi,E)$ is set to zero. The value λ_D is modified in accordance with the mesh spacing as described for linear screening (above).

Frozen Ion

The "Frozen Ion" treatment is intended for short timescale (typically a few microseconds or less) problems for which it is a good approximation to assume that electrons are in barometric equilibrium, but ions have not moved. The space charge function depends on the mesh-dependent potential, ϕ_1 which satisfies,

$$1 - \exp(\phi_1 / \theta) = -(\lambda / D)^2 (\phi_1 / \theta) \tag{27}$$

Where D is the local mesh spacing divided by the DEBLIM parameter (ϕ_1 is zero D≤λ.) The space charge is then given by,

$$\rho / \varepsilon_0 = \begin{cases} (\phi_1 / D^2)(1 - \exp(\phi / \phi_1)) & \phi \geq 0 \\ -\phi / D^2 & 0 \geq \phi \geq \phi_1 \\ -\phi_1 / D^2 + (\theta / \lambda^2)(\exp(\phi_1 / \theta) - \exp(\phi / \theta)) & \phi_1 \geq \phi \end{cases} \tag{28}$$

Special

Developer user only

Full Trajectory Ions

Cell-centered ion densities are taken from RHO_Elec and node ion charges from RHO_ION. The Traj_Ion DLL is used to convert RHO_Elec to RHO_Ion and fill in special cell Rho_Elec's, which are not calculated by Tracker, using RHO_GI. Electrons are barometric.

Hybrid-PIC

This algorithm is used for timescales on which it is practical to treat ion motion, but electrons are considered in barometric equilibrium. The ion density has been stored in the RHO_Ion array by the Tracker module, and the electron charge density must be added. Subroutine RhoOfP returns the electron charge density as,

$$\rho / \varepsilon_0 = \begin{cases} -(\phi + \theta(L / \lambda)^2 / L^2) & \phi \geq 0 \\ -(\theta / \lambda^2) \exp(\phi \lambda^2 / \theta L^2) & \phi \leq 0 \end{cases} \tag{29}$$
$$L = \max[L, D]$$

Full-PIC

For this option, it is assumed that the Tracker module has stored both the electron and ion charge densities in the RHO_Ion array.

When **Geometric Wake Initialization** is checked, the wake of the (uncharged) spacecraft surfaces is calculated. ("Neutral Approximation.") The velocity is specified on the environment page. The calculation is only done in a "NEW" potential run (iteration 0). The mass used is specified by the mass of the selected species.

The **Target Average (RMS) Error** gives the value of the "RMSERR" parameter below which the potential is considered converged.

The Advanced button brings up the Advanced parameters screen.

The Iteration box specifies how many times the potentials and particle trajectories are iteratively calculated. The fraction old potential specifies how much of the solution from the previous major iteration is mixed with the solution of the current major iteration. The fraction old density does the same for the charge density computed by the particle tracking.

2.9.2   Advanced Parameters

Advanced users have a variety of options at their disposal to control the potential solver. Figure 23 below shows the available parameters that appear on the "Advanced Potential Solver Parameters" window once the "Advanced" option is chosen. A description of all parameters is provided below.

Figure 23. Potential Solver Input Screen.

Debye Scaling: The 'DebLim' parameter (see below) may be applied based on the local grid spacing or on the primary grid spacing.

Timer Level and Diagnostics: These parameters govern optional printout from various portions of the potential solver.

Grid: It is possible to apply the potential solver to a subset of the *DynaPAC* grids.

Convergence Criteria:

Normally only the two parameters "Maxits" and "RMSmin" are varied from their default values.

Maxits - The maximum number of major, or "space charge" potential iterations to be performed.
RdRmin - The value of the "RDOTR" parameter below which the potential is considered converged.
Maxitc - The maximum number of minor iterations within each conjugate gradient solution.
PotCon - The number of orders of magnitude that the RDOTR drops within each major iteration before it is considered converged.

30

DebLim - The number of Debye screening lengths allowed per zone. The various space charge treatments limit the amount of space charge in a zone in accordance with this parameter.
Conv. Effect - Whether the analytic trajectory convergence treatment is used in the NON_LINEAR problem type.

The wake parameters are the same as those used by the *POLAR* computer code.

2.9.3 *Correspondence with* DynaPAC *Keywords*

Table 6 gives the correspondence between *DynaPAC* keywords and *Nascap-2k* screen values.

Table 6. Potentials in Space Input File Contents

DynaPAC keyword	Tab	Comments
RUN		Use "New" for first potent call and "Continue" for rest
SOLUTION_MIX	Potential	The value is labeled fraction old potential and varies for different input files. The first potent call uses 0.0.
TEMP	Environment	
DEBYE	Environment	
DENSITY	Environment	
OBJVEL	Environment	
PROBLEM	Potential	Charge density model Hybrid PIC = Track_Ion Full PIC = Explicit others obvious
RMASS	Potential	
RMSMIN	Potential	
WAKE	Potential	
CONV_EFFECT	Advanced	
DEBLIM	Advanced	
DEBYE_SCALE	Advanced	
DIAGFINAL	Advanced	
DIAGINIT	Advanced	
DIAGINTERFACE	Advanced	
DIAGMATRIX	Advanced	
DIAGSCG	Advanced	
DIAGSCREEN	Advanced	
DIAGWAKE	Advanced	
GRIDHIGH	Advanced	
GRIDLOW	Advanced	
MAXITC	Advanced	
MAXITS	Advanced	
NADD	Advanced	
NPHI	Advanced	
NTHETA	Advanced	
POTCON	Advanced	
RDRMIN	Advanced	
TIMER	Advanced	
ALGORITHM		Not used

31

2.9.4 *Potential Interpolation Scheme*

2.9.4.1 Continuous Field Interpolants

We originally proposed using triquadratic interpolants in *DynaPAC* in order to obtain more nearly continuous electric fields than were provided by the trilinear interpolants used in previous three dimensional codes. Experience has shown that the proposed method suffers from the two disadvantages that (1) the fields are still not strictly continuous; and (2) the difference in weight between the corner nodes and the edge-center nodes leads to a poorly conditioned matrix. Therefore, we have abandoned the triquadratic formulation in favor of a finite element interpolation scheme that has strictly continuous electric fields.

The basic interpolation functions consist of functions that have unit value and zero slope at their home nodes, and zero value and slope at opposite nodes:

$$F_0 = (z-1)^2/(1-2z+2z^2)$$
$$F_1 = z^2/(1-2z+2z^2)$$

(30)

and functions that have zero value and unit slope at their home nodes and zero value and slope at opposite nodes:

$$G_0 = z(1-z) \, F_0$$
$$G_1 = z(z-1) \, F_1$$

(31)

The functions F_0 and G_0 are shown in Figure 24. It can be verified that these functions can exactly reproduce constant, linear, and quadratic potentials.

We generalize to three dimensions by assigning to each node of the unit cube four interpolation functions corresponding to the potential, x-component of potential gradient, y-component of potential gradient, and z-component of potential gradient for that node. Thus, the contribution of these four quantities at point [0,0,0] to the potential at [x,y,z] is governed by the interpolation functions $F_0(x) \, F_0(y) \, F_0(z)$, $G_0(x) \, F_0(y) \, F_0(z)$, $F_0(x) \, G_0(y) \, F_0(z)$, and $F_0(x) \, F_0(y) \, G_0(z)$. (Contributions of other cube corners are obtained by changing "0" subscripts to "1" as appropriate.) We elect to omit terms of the form GFG or GGG.
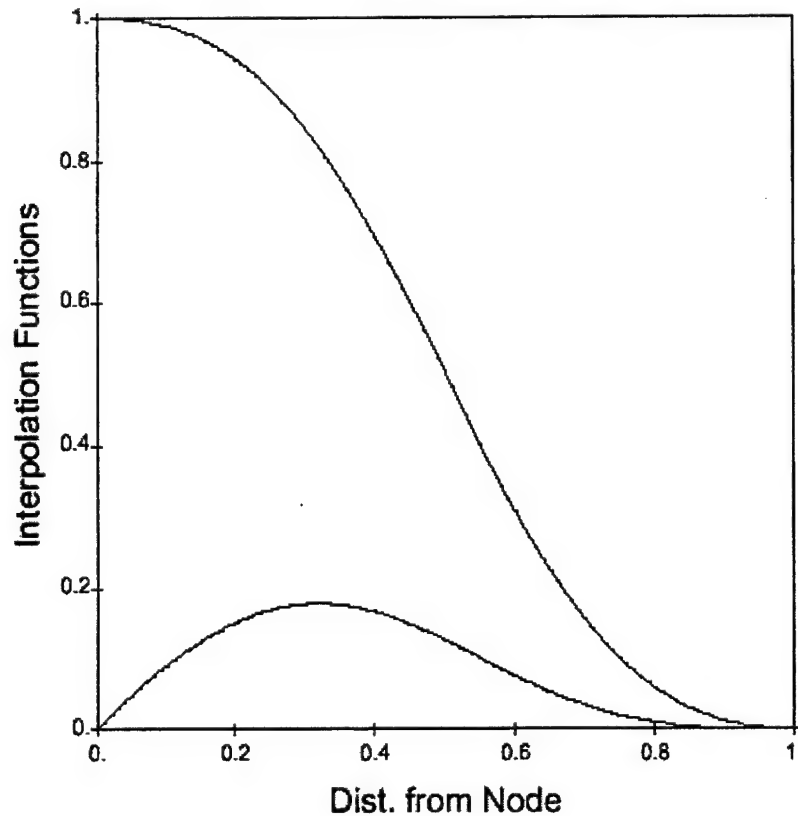
Figure 24. Interpolation Functions F0 and G0.

2.9.4.2   Interpolating Potentials and Fields for Special Elements

Special elements are those elements which contain surfaces. Because surface-containing elements are not empty cubes, the potential interpolation functions described in the previous section cannot be straightforwardly applied. To develop finite element matrices for these cells, we require a prescription for calculating the potential and electric field in the cell interior. Note that a "special element" is bounded by three types of surfaces: (1) square surfaces bounded by grid edges and shared with adjacent (presumably empty) elements; (2) object surfaces; and (3) surfaces bounded by both object points or edges and grid points or edges. On type (1) surfaces we must use the potential and field interpolation described in the previous section. On object surfaces, the potential and electric field must be expressed in terms of the object's surface cell potentials and normal fields. Type (3) surfaces must smoothly blend between the two. We describe below an algorithm to express the potential and field at any point in the cell volume in terms of the grid point potentials and fields, and the surface cell potentials and normal fields. The algorithm has the property that, when applied to a cell with six type (1) surfaces, constant, linear, and quadratic potentials can be represented exactly.

Let **R** be a point in a volume bounded by a set of surfaces {S}, each of which may be a triangle or a planar convex quadrilateral. For a given S, let **P** be the point on S nearest **R,** let $\phi(P)$ and

33

$E(P)$ be the potential and electric field at P, and n be the unit normal (from the surface point P into the volume) to the surface. Let

$$\begin{aligned} \mathbf{d} &= \mathbf{R} - \mathbf{P} \\ d^2 &= \mathbf{d} \bullet \mathbf{d} \\ K(S) &= A(S)/d^2 \quad for\ \mathbf{d} \bullet \mathbf{n} > 0 \\ N &= \sum_{\{s\}} K(S) \end{aligned} \tag{32}$$

Let l be the distance from P in direction n to the next surface intersection. (In practice, it is adequate to extend l to the intersection with the surface of the rectangular parallelepiped bounding the volume.) Then the contribution of S to the potential at $\mathbf{R}$ is

$$\phi_S(\mathbf{R}) = (K(S)/N)\big[\phi(P) - (1 - \mathbf{d} \bullet \mathbf{n}/l)(\mathbf{d} \bullet \mathbf{n})(E(P) \bullet \mathbf{n})\big] \tag{33}$$

and the total potential is found by summing the contributions from all the surfaces.

The electric field is found by differentiating the above. The contribution of S to the electric field is

$$\mathbf{E}_S = \phi_S \big[\nabla N/N - \nabla K(S)/K(S)\big] + (K(S)/N)\big[-\nabla \phi(P) + (1 - 2\mathbf{d} \bullet \mathbf{n}/l)\mathbf{n}(E(P) \bullet \mathbf{n})\big] \tag{34}$$

Where $\nabla \phi(P)$ is taken tangential to the surface, so that

$$\begin{aligned} \nabla \phi(P) &= 0 & \text{for } \mathbf{P} \text{ at a vertex of S} \\ \nabla \phi(P) &= -\mathbf{e}(\mathbf{e} \bullet E(P)) & \text{for } \mathbf{P} \text{ on edge w/direction } \mathbf{e} \\ \nabla \phi(P) &= -E(P) + \mathbf{n}(E(P) \bullet \mathbf{n}) & \text{for } \mathbf{P} \text{ in interior surface} \end{aligned}$$

To implement this we interpolate surface fields and potentials on bounding surfaces in a manner that maintains the continuous field property. Potentials and normal field components are defined at the centroids of the original object surfaces and area-weighted averaged at surface corners. Potentials, normal fields, and tangential field components at all surface points that are vertices f bounding surfaces are expressed as linear combinations of centroid potentials and fields. This transformation is given by the matrix TNdCnt in the special element's "Bound_Surfs"record. The format of the "Bound_Surfs" record is given in Table 7.

Table 7. Contents of Bound_Surfs Record

Word	Variable	Contents
1	NCSurf	Number of bounding polygons
2	NCNode	Number of nodes(a)
3	NCnts	Number of centroids
4	LTrans	Length of constraint relation matrix
next 8*NCSurf	JCSurf(8,*)	Bounding polygon node list
next 3*NCNode	RCNode(3,*)	Node locations
next NCnts	MatCnt(*)	Centroid list
next LTrans	LTCnt(*)	Constraint relation map
next LTrans	TNdCnt(*)	Constraint relation matrix

34

Nodes 1-8 are cube corner nodes. Remaining nodes are surface cell centroids, surface cell corners (constrained), and additional surface nodes (constrained).

2.10 Particle Tab

Macroparticles may be used in *Nascap-2k* for a number of purposes including

1) Studying and/or displaying representative particle trajectories;
2) Calculating surface currents arising from sheath currents;
3) Studying wake structure;
4) Calculating steady-state, self-consistent charge densities;
5) Calculating time-dependent charge densities and surface currents.

Particles can be generated throughout the volume, along a sheath surface, along a contour line, in accordance with external input, or along magnetic field lines. The particle tracker computes the motion of all or a subset of the particles for a maximum time or grid cycling, recording surface currents, and (optionally) plotting trajectories, accumulating steady-state charge density, or calculating new charge density at the updated time. After the particle tracker is run, the particle files are left with updated particle positions and velocities, and the time and cycle number is updated in the DataBase.

The Tracking tab has three subtabs (see Figure 25 to Figure 27), which are enabled depending on the choice of Problem. The fourth subtab is used to specify tracking parameters for trajectories displayed on the Results 3-D tab.

2.10.1 Specifying the Initial Particle Distribution

All versions of this subtab have the exclusive choices of Sheath, B-field, Boundary, and None of the above.

Sheath: Generate particles representing sheath currents. It is accompanied by a placeholder to enter the sheath potential.

B-field: This option is used to generate particles where magnetic field lines enter the computational space. The subscreen is used to enter the particle energy, the (square root of the) number of particles for each boundary surface, and the magnetic field.

Boundary: Generate a thermal distribution of particles at the problem boundaries (more in section 2.10.2)

All versions of this subtab also have the option of adding an external file. The user has the option of typing in the filename or browsing to the file. The particle species to be used with the external file is shown on the right hand side of the screen.

For time-dependent problems (Figure 27), the user also has the option of specifying that a uniform distribution should be used for initialization.
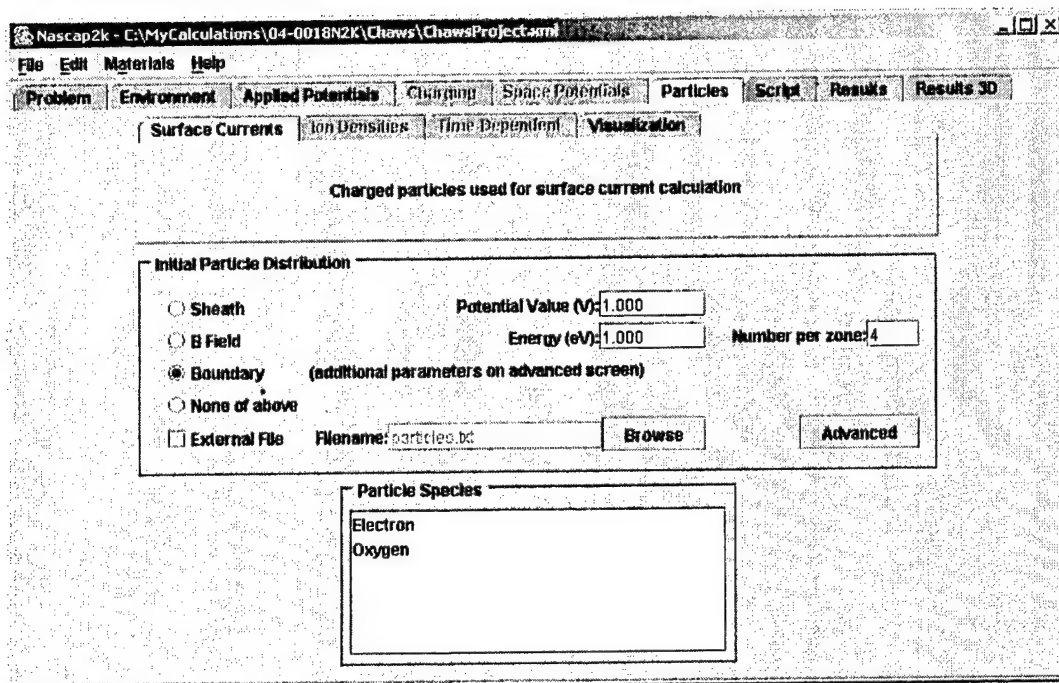
Figure 25. Particle Generation and Tracking Subtab for Surface Currents.
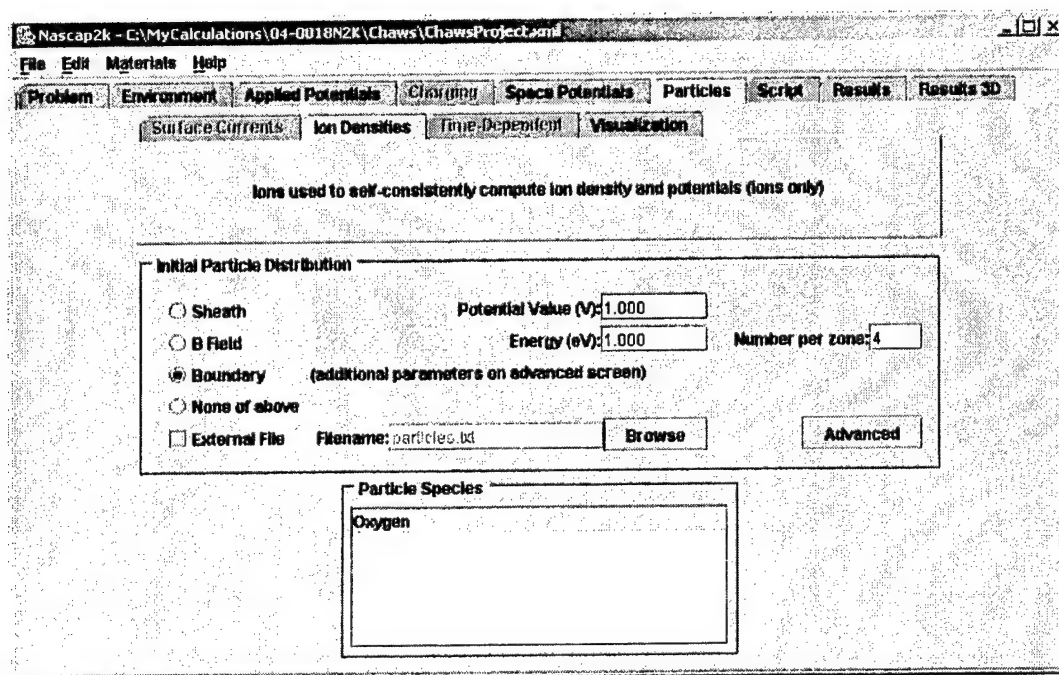
Figure 26. Particle Generation and Tracking Subtab for Self-consistent with Ion Trajectories.

36
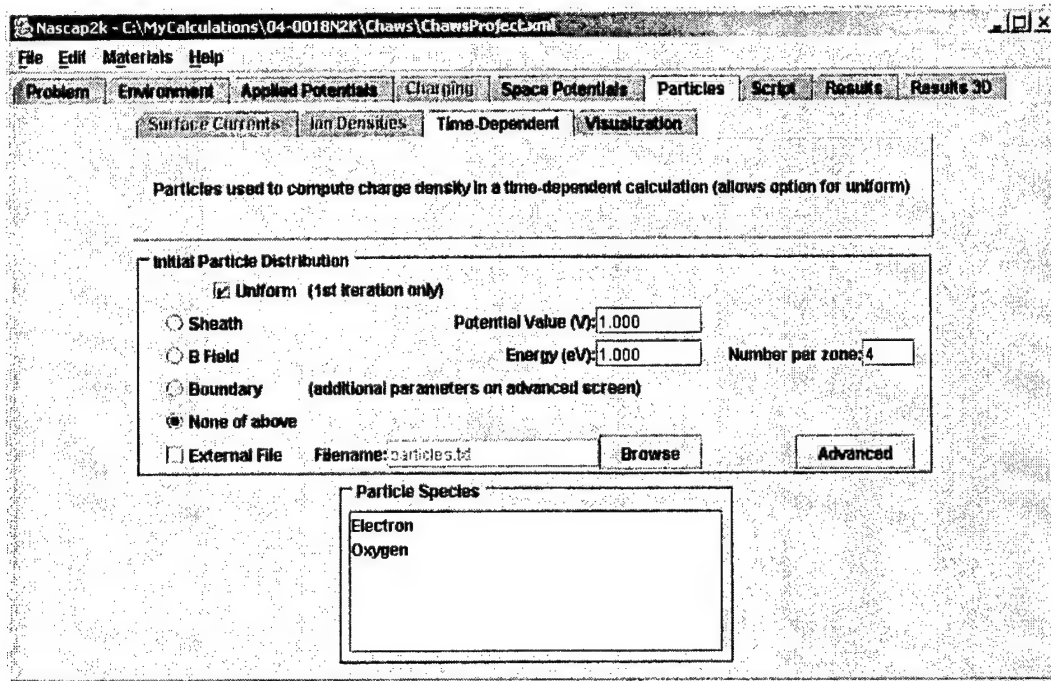
Figure 27. Particle Generation and Tracking Subtab for Time-dependent Plasma.

2.10.2 Particle Generation and Tracking Advanced Choices

There are two main sections, Particle generation and Tracking. Under Particle generation there are two subsections, Diagnostics and Boundary particle parameters. The Boundary particle parameters are for particles generated with a Boundary initial particle distribution. Under Tracking there are two subsections, General and Diagnostics. All of the diagnostics parameters are pull downs with the available values shown in Figure 28.

The **diagnostic level** numbers specify the amount of diagnostic output sent to the output files.

The **timer level** specifies the frequency of cpu time monitoring.

Tracking Time per timestep The maximum time [seconds] a particle is to be tracked. For time dependent problems, the time variable is incremented by this amount.

Maximum number of timesteps The maximum number of substeps per particle per iteration.

Distance a particle can go in one timestep The maximum distance (in local grid units) a particle moves during a substep. Substep distances may be far less than this value. Electrons gyrating in a magnetic field move only for a fraction of the cyclotron period. Slow ions may move less than "Max Dx" in the tracking time.

MaxIter The maximum number of cycles through the list of grids.

37

Save to files every How often potentials are saved (by subroutine spaini.F) in time-dependent problems.

Tracking limits Limits (in primary grid units) of initial particle locations. Particles originating outside these limits are ignored. If not set (all zeroes) all particles are tracked.

Figure 28. Particle Generation and Tracking Advanced Screen.

2.10.3 Visualization Subtab

The particles subtab for visualization is shown in Figure 29. This tab has an additional option for specifying the initial particle distribution. "Contour" specifies that macroparticles are to be generated at the intersection of a cut-plane and a constant potential surface (such as a sheath). Either trajectories or particle present positions can be specified. The plot limits are the limits within which particles or trajectories are plotted. If not set (all zeroes) no plot is produced.

The plot is displayed on the Results 3-D tab.

Figure 29. Particles Subtab for Visualization.

2.10.4 Correspondence with DynaPAC Keywords

Tables 8 and 9 give the correspondence between *DynaPAC* keywords and *Nascap-2k* screen values. Each species has its PartGen input file

Table 8. Contents of PartGen Input File.

DynaPAC keyword	Tab	Comments
PART_TYPE	Tracking	Allowed values are "Sheath," "Contour," "B_field," "Boundary," "External" For Time Dependent problems, the first call will specify "Default" and all others something else if the checkbox next to uniform is checked
SHEATH_POT	Tracking	Only for Sheath and Contour type
ENERGY	Tracking	Only for Bfield type
N_ZONE	Tracking	Only for Bfield type
CUT_PLANE	Tracking	Only for Contours
CUT_PLANE	Tracking	Only for Contours
EXTERNAL_FILE	Tracking	Only for external file
EXTERNAL_TYPE	Tracking	(Default to formatted)
B_FIELD	Environment	
TION	Environment	Use plasma temperature
RHOION	Environment	Use plasma density * % of particle
VRAM	Environment	
SPECIES	Environment	
PVX	Advanced	Only for Boundary type
PVY	Advanced	Only for Boundary type
PVZ	Advanced	Only for Boundary type
DIAG_PARTGEN	Advanced	
DIAG_DYNALIB	Advanced	
TIMER	Advanced	
		Not Used
SUBDIVISION	Advanced	

40

Table 9. Input File Contents for Tracker

DynaPAC keyword	Tab	Comments
PROCESS	Problem or Results 3D	For currents and visualization use "Trajectories" except when "Plot particles" is specified (visualization only) For self-consistent with ion trajectories use "Traj_Charge" For time dependent problems use "Space_Charge"
B_FIELD	Environment	
TRACK_TIME	Tracking	
SPECIES	Tracking	
MAX_STEP	Tracking	
MIX	Iteration	The value is labeled fraction old ion density and varies for different input files. Handle the same as in the discussion above for the corresponding potentials in space input parameter.
DX_MAX	Advanced	
MAX_ITER	Advanced	
SAVE_INTERVAL	Advanced	
SAVE_INTERVAL	Advanced	
X_LIMIT, Y_LIMIT, Z_LIMIT	Advanced	
X_PLOT_LIMIT, Y_PLOT_LIMIT, Z_PLOT_LIMIT	Tracking	If not visualization, set to all zeros.
DIAG_TRACKER	Advanced	
DIAG_DYNALIB	Advanced	
TIMER	Advanced	
		Not Used
TITLE		Not Used
CUT_PLANE		Not Used
CUT_PLANE		Not Used
ALGORITHM		Not Used

2.10.5 External File Format

The format of the external file from which particle data may be read is as follows.

```
POSITION    x       y       z
DIRECTION   dx      dy      dz
ENERGY      energy
```

where the position is in primary grid units, the direction need not be normalized, and the energy is kinetic energy in electron volts. Alternatively, the keyword E_TOTAL may be used to give the total particle energy. Each such three card sequence results in definition of a particle. A unit particle weight is assigned, on the assumption that this input is used only for trajectory plotting purposes.

To specify a large number of "EXTERNAL" particles, you may use an unformatted file. The file is read in subroutine REDEXT with the statement

    Read (IExtFi,end=910) PrtPos,Direc,EKin,ETot,PrtWgt

Any non-negative kinetic energy is considered valid. Otherwise, the total energy is considered valid. The meaning of the PrtWgt varies depending on the application, but is most commonly in amperes.

2.10.6 Particle Generation Physics and Numeric Background

In this section we briefly describe how the particle generator operates for each "particle type," pointing to some of the key subroutines in the process. In general, subroutine inivel.F is used to assign initial velocities to particles. Subroutine convrt.F converts volume weights to charge or current weights.

For the "SHEATH" particle type, particle generation is under the control of subroutine genpa2.F. Actual particle generation is done by subroutine sthpar.F. SthPar varies its procedure if magnetic effects dominate, as determined by comparing the Larmor radius to the outer grid spacing. Subroutine dozone.F determines the particle positions and area weights, also eliminating high field or bipolar zones. For non-magnetic cases, subroutines inivel.F and dflux.F determine particle velocity and current. For magnetic cases subroutine magsth performs this function, assigning a "$\cos \theta$" factor to account for particles traveling along field lines to reach the sheath surface. Particle weights are amperes.

For the "B_FIELD" particle type particle generation is under the control of subroutine genpa5.F. Particles are generated only on the surface of the outermost grid. Actual particle generation is done by subroutine bparts.f. Particle weights are amperes.

For the "CONTOUR" particle type particle generation is under the control of subroutine genpa3.F. Actual particle generation is done by subroutine cntpar.F.

2.10.7 Tracking Physics and Numeric Background

The particle tracker operates on the *prefix*.PTn files, which are organized in pages of 1000 particles each. The code keeps track of how many particles on each page belong to each grid. An "iteration" consists of looping through each grid, and within that through the particles belonging to that grid. Subroutine trkpar.F supervises the tracking of each particle by subroutine movpar.F. Each particle is tracked until one of the following conditions occurs:

1) The particle strikes the spacecraft;
2) The particle is found in a grid other than the current grid or its parent, in which case it is transferred to the new grid;
3) The particle exits the computational space;
4) The trajectory time reaches the requested particle tracking time;
5) The number of substeps exceeds the maximum substep number;
6) For the TRJ_CHARGE process, if the particle is found outside the "sheath," (i.e., in a region where it has very low kinetic energy).

Trajectories stopped for conditions (2) or (5) above are picked up during the next iteration.

After processing each grid Tracker prints the particle number and total weight for each of several particle categories:

1) "new particles" is the set of particles with which Tracker started;
2) "partially tracked" refers to particles which have not hit the spacecraft or left the primary grid. These particles may or may not yet have been tracked for the requested particle tracking time.
3) "dead" particles are those which struck the spacecraft.
4) "went off primary grid" counts those particles which have exited the computational space.
5) "trapped" is presently a null category, as there is no criterion for trapped particles. Trapped particles appear as "partially tracked."
6) "unknown status" is another category of particles with no known means of identification.

When Tracker is finished with the requested number of iterations, it prints the total current to the spacecraft and a table apportioning that current by material and conductor. Also, a "Hit" file is printed listing each particle that struck the spacecraft with its vital statistics which may be used for further processing. The print and format statements for this file may be found in subroutine wripar.F.

2.11 Script Tab

The "Script" tab is where the calculation is actually executed. There are two subtabs, "Run Script" and "Edit Script." All previous tabs give the user the opportunity to set up the problem. By doing so the user also specifies a default script. Pressing the "Build Script" button generates the default script. For problems other than the standard ones, the "Edit Script" tab allows the user to modify the script. Pushing the "Run Script" button runs the calculation specified by the script and all the parameters specified on the various pages. The "Save Files" button writes out all the

input files of the various modules without actually running the code. If execution is prematurely interrupted, either accidentally or on purpose, the files remain in a stable state and execution can be restarted later.

2.11.1 Edit Script Subtab

Figure 30 (RHS) shows a typical script for a interplanetary charging calculation. It consists of commands, (preceded by a capital "C") folders, and attributes (preceded by a capital "A"). A list of all available commands is shown on the left hand side of Figure 30. The user may "Add Commands" to the script from the list, "Delete Items" and/or "Duplicate Items." For example, let us assume that the user wishes to change the end time for the charging calculation from 30 sec to 1000 sec. To do so, click twice on the line: A endtime 30.0 and modifies directly the number 30 to 1000. (Note that this script lacks a ReadPhotoemission command, so that default photoemission will be used and positive charging underestimated.)
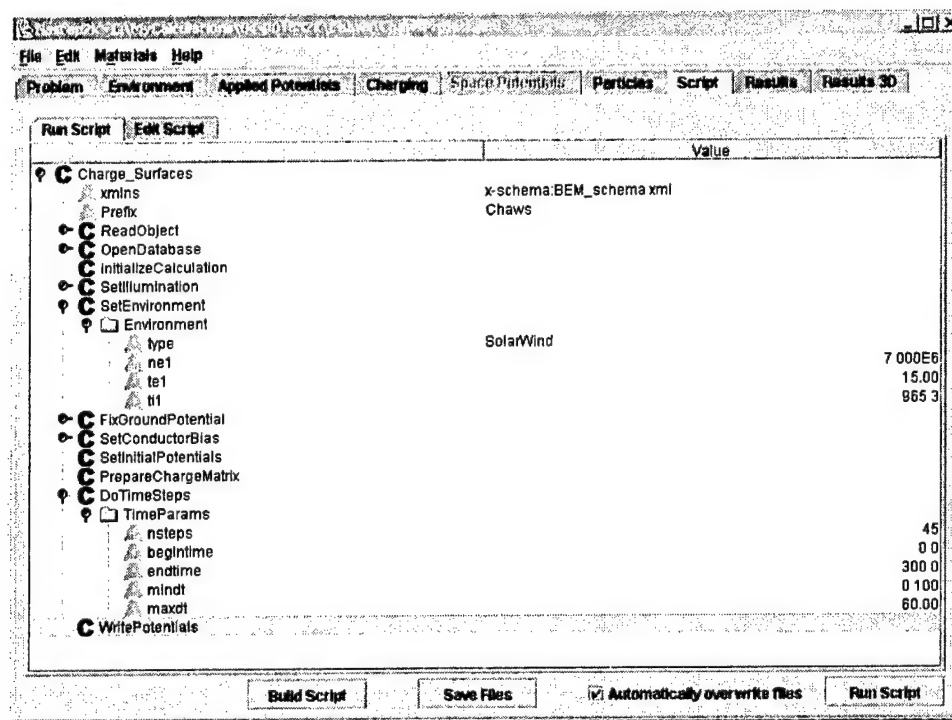
Figure 30. The "Edit Script" Tab: Making Problem Changes Directly by Modifying the Script.

2.11.2 Charge Surfaces Commands

The following commands to the BEM module (Charge surfaces) can be added, deleted, or edited.

AppendObject - appends object from XML info to object in existing Database
      Attributes: FileName - path to xml object file containing nodes and elements
            x - x-offset (meters) of appended object
            y - y-offset (meters) of appended object

44

z - z-offset (meters) of appended object
ChildElements: None

Calculate_Matrices - Calculate the BEM matrices for the model.
Attributes: None.
ChildElements: None.

DefineInsulators - Step towards eliminating conductive surfaces from the matrices.
Attributes: None.
ChildElements: None.

DoTimeSteps - Perform timestepping of charging and potentials.
Attributes: None
ChildElements: TimeParams - (at least one required.)

FixConductorPotential - Fixes conductor potential to a specified value
Attributes: Value - value of ground potential
Index - Index of conductor.
ChildElements: None

FixGroundPotential - Fixes ground potential to a specified value
Attributes: Value - value of ground potential
ChildElements: None

InitializeCalculation - Performs initialization needed for BEM Charging calculation:
1. Attempts Read_Matrices command.
2. If matrices not read:
Calculate_Matrices
Write_Matrices
3. DefineInsulators
4. PrepareChargeMatrix

PrepareChargeMatrix - Compute additional matrices needed for timestepping.
Attributes: None.
ChildElements: None.

Read_Matrices - Read in the matrices for the model.
(Reads in capacitancematrix, fieldmatrix, DMatrix.)
(Substitute this for Calculate_Matrices if the matrices have been written out.)
Attributes: None.
ChildElements: None.

OpenDatabase - Opens Database and gets object info
Attributes: Prefix (optional)
- prefix to use for Database files
ChildElements: None

ReadPhotoemission - Read photoelectron spectral data
    Attributes: FileName (optional)
            (if no FileName, looks for "prefix.photo.xml."
             If that is not found, uses default photoemission.)
    ChildElements: None

ReadPotentials - Read potentials from the Database.
    Attributes: None.
    ChildElements: None.

ReadObject - reads object from XML info to Database
    Attributes: FileName (optional)
            - path to xml object file containing nodes and elements
    ChildElements: None

Set_Element_Potentials - set potential of all elements to a single value.
    Attributes: Value (defaults to 0.0)

SetConductorBias - Sets bias value for a conductor.
    Attributes: Value - value of bias potential.
            Index - index of biased conductor.
            Index2 - index of reference conductor.

SetCustomCurrentDLL - Specifies that element currents are to be obtained from a custom DLL
    Attributes: FileName - name of custom DLL
            - Custom DLL contains the function *extern "C" __declspec(dllexport)*
CALLBACK getCustomCurrent(element* elem, double* I0, double* I1); which calculates the
current I0 and its derivative I1 (currents / epsilon0) from the properties of the element.

SetEnvironment - Sets an environment
    Attributes: None
    Child Elements: Environment - (at least one required.)

SetIllumination - Set sun intensity and direction
    Attributes: Value - ratio to solar intensity at Earth's orbit.
            x, y, and z - components of sun vector.
    ChildElements: None.

SetInitialPotentials - Set element potentials to their initial values.
    Attributes: None.
    ChildElements: None.

Write_Matrices - Write out the matrices for the model.
        (Writes out capacitance matrix, field matrix, DMatrix.)
    Attributes: None.

ChildElements: None.
WritePotentials - Write potentials to the Database.
    Attributes: None.
    ChildElements: None.

Child Elements

Environment - Geosynchronous or other Environment Description
    Attributes: ne1, te1, ni1, ti1 (required) - properties of primary maxwellian;
        ne2, te2, ni2, ti2 (optional) - properties of secondary maxwellian;
        from, to (optional) - effective time for time-dependent environment
        type - required if child of SetEnvironment command
            Currently supported are "GEO," "SolarWind," "Custom"
            For "SolarWind" ti1 is treated as the ion energy (about 1000 eV), and
                ne2,te2,ni2,ti2, from, to are ignored.
            For "Custom" only attribute is "FileName," which is name
                of custom DLL to use for charging currents.

TimeParams - Parameters for setting timesteps
    Attributes: (all optional)
        begintime - timestamp (s) at beginning of timestep sequence
                (default 0 or previous endtime).
        endtime - timestamp (s) at end of timestep sequence (default 1).
        nsteps - number of timesteps (default 1).
        mindt - duration of first timestep (default 0.1).
        maxdt - maximum allowed timestep duration (default is no maximum).

2.11.3 Operation of the Potential Solver

In this section we discuss the operation of the potential solver from the point of view of monitoring its progress. In *Nascap-2k* the "Potent Monitor" is shown during a calculation of potentials in space.
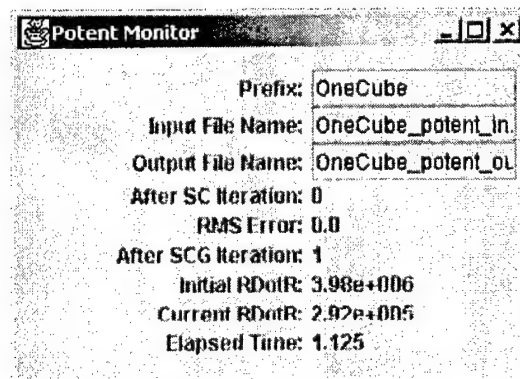

Figure 31. Screen for Monitoring the Potential Calculation.

47

A potential solver run consists of an initialization phase, a number of major ("space charge") iterations of the potential solution, and a brief exit phase. In the initialization phase, the input parameters are read and echoed, and the DataBase information is processed. Grid information (mesh size and sheath potential) is printed. The grid interface pairs list is formed and printed. Conductor potentials are printed.

At the beginning of each major potential iteration the space charge function and its derivative are evaluated cell by cell, and the conjugate gradient process is initialized. The "initial RDotR" is a measure of the current error in the potential solution. For most cases the "initial RDotR" should decrease monotonically beyond the first few major iterations.

During the conjugate gradient process (governed by subroutine psscg.F) the "RDotR" parameter is printed for each minor iteration. This parameter should generally decrease, but for most cases does not decrease monotonically. The conjugate gradient process is deemed converged when the criteria discussed above (usually two orders of magnitude decrease in "RDotR") are satisfied.

We are now ready to conclude the major iteration. The most time-consuming task is to calculate and update surface electric fields. This requires a full matrix operation (performed by subroutine ecoprd.F), as the electric field value is related to the residual for the corresponding potential. The field updates (subject to several restrictions) are performed by subroutine eupdat.F. The new surface potentials and fields are printed. The difference between the new and previous potential solutions are expressed as root-mean-square errors, and are printed grid by grid ("grep RMS ps_out") and overall ("grep rmserr ps_out"). If the root-mean-square errors remain constant, solution-mixing may ameliorate or solve the problem. (A particularly large "RMS Error" for a particular grid may or may not indicate a problem in obtaining a solution.)

The Potential Solver is ready to conclude when the requested number of major iterations have been performed, or when the "RMS err" has been reduced below its minimum value. By default, the central Z-slice potentials and electric fields are printed for each grid. In these printouts three values are printed for each grid point: the potential value appears at the print position corresponding to the grid point; the x-direction potential gradient [volts per meter] is printed to the right of the grid point, and the y-direction potential gradient above the grid point. The z-direction potential gradient does not appear.

3. NASCAP-2K OUTPUT: VIEWING RESULTS

3.1 Results Tab; Time Dependence

The "Results" tab is for analysis of potential, electric field and current to surfaces as a function of time. Plots may be generated for groups of surfaces, single surface elements, and conductors using the top, middle and bottom sections, respectively, on the left hand side of Figure 32. The plotted results are also available on the "Text" subtab for obtaining specific information or to import to another plotter or analysis tool. To better illustrate the plotting capability a specific charging problem is described below.

In Figure 32 the user has chosen to plot the potential of all the conductors and materials as a function of time. The orientation limitation is specified by a vector and an angle, with an angle of 180° to include all orientations. The dot product of the vector and the surface normal must be less than the cosine of the specified angle. Because all surfaces are included in this specification, the minimum, maximum and average potential values of the spacecraft surfaces are displayed. By checking the second row, the user would analyze only those cells pointing away from the sun direction. The third row would produce min, max and average potential values for all sunward-facing OSR surfaces.
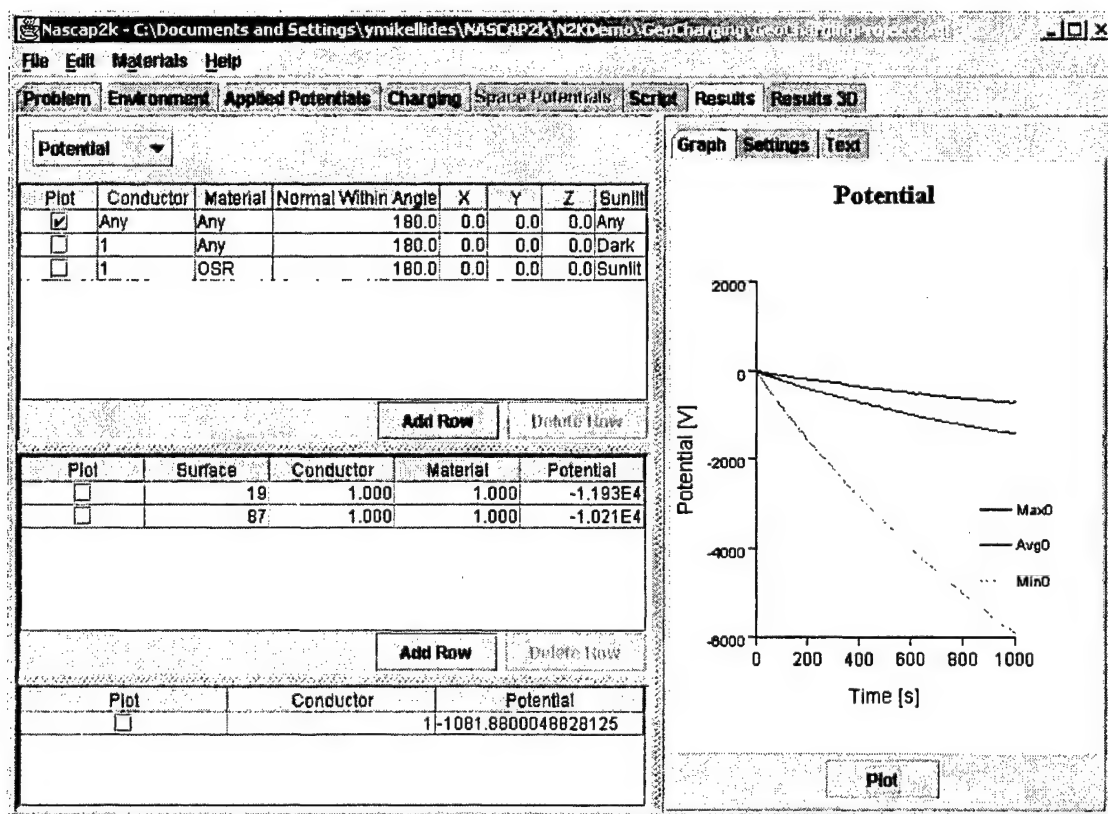
Figure 32. Results Screen: Plotting Minimum, Maximum and Average Values of Conductors and/or Materials.

49

Time histories of specific surfaces are plotted using the middle section of the screen as shown in Figure 33. In this example the potential history of surfaces #37 and #62 (see Figure 34) are shown. Note that the third and fourth columns automatically display the corresponding conductor number and material. The last column displays the value at the end of the calculation, in this case at t=1000 sec.

Finally the bottom screen may be used to plot values for a specific conductor. The value shown in the last column is the value at the final timestep.
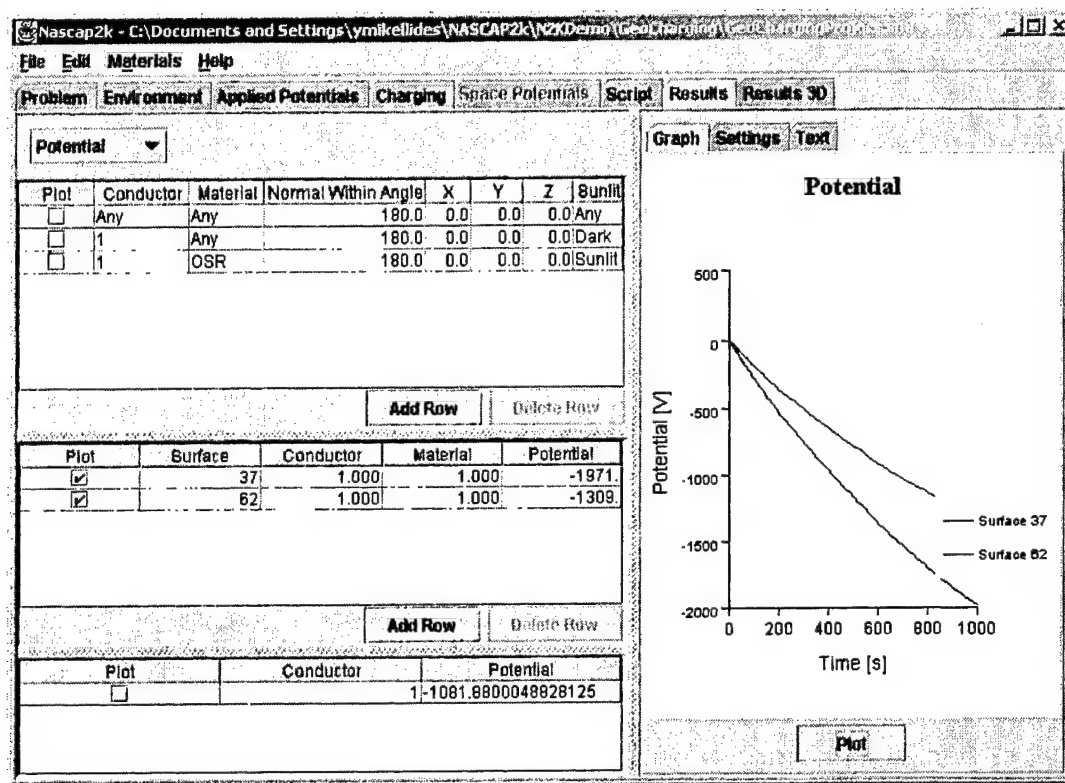
Figure 33: Results Screen: Plotting Values at Surfaces.

The "Settings" subtab can be accessed to control the axes scales and legend while the "Text" subtab provides a numerical list of the values that are being plotted, and can therefore be copied and pasted into other graphics programs for further manipulation.
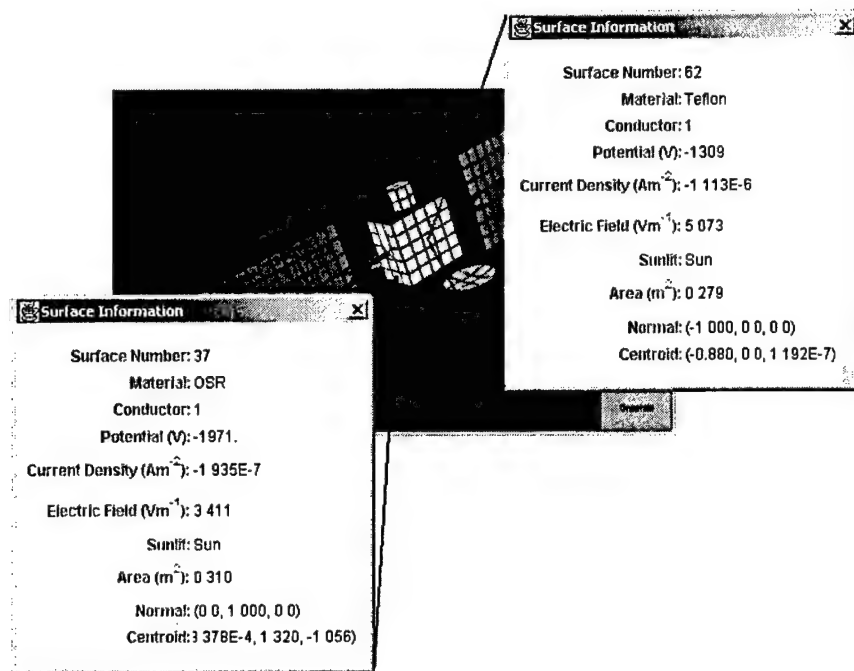
Figure 34. Identifying Surface Information.

3.2 Results 3-D Tab

The "Results 3D" tab displays results in two and three dimensions. Initially, "Results 3D" displays potentials on the surfaces of the object. Additional options include "Material," "Conductor" and "Electric Field" (Figure 35). As in Object Toolkit, the object may be viewed from a variety of angles and positions using the "Cursor Tools" and "Direct Movement and Rotation" buttons. As shown in Figure 34 and Figure 36, information associated with a specific element on a surface may also be displayed using the "Element Info Tool."

Figure 36 shows the cutplane capability in the *Nascap-2k* GUI. The underlying coding is from *DynaPAC*'s Scanner. The cutplane is specified as normal to an axis and at a position measured in meters from the center of the grid. The underlying structure allows multiple cutplanes to be plotted, although only a single cutplane can be specified in the GUI. Note that running a script deletes the cut-plane plot object and the trajectory plot object.

Checking the trajectories checkbox, specifies that trajectories of selected particles are to be computed and shown. Choosing the "Select Trajectories" button takes the user to the Visualization subtab of the Particles tab, where the selected particles are specified.
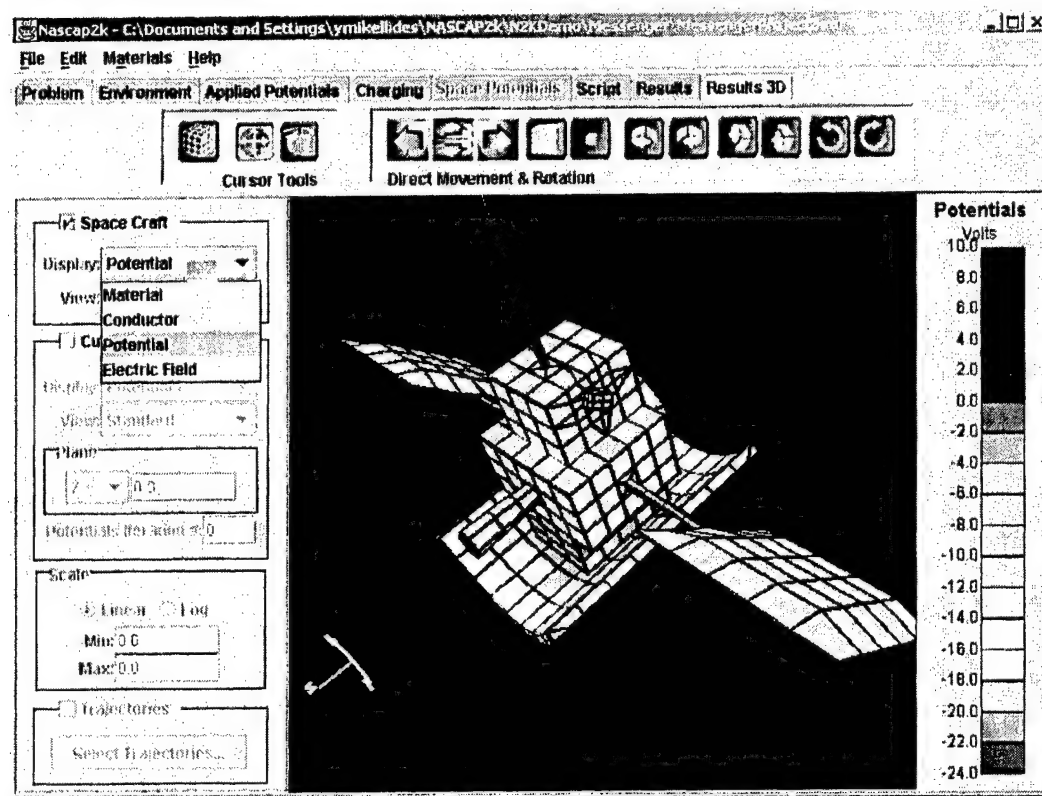
Figure 35. Results 3D: Displaying Results on the 3D Object.

Figure 36. Results 3D: Displaying 2-D Results in Cutplanes.

52

Figure 37. Particle Trajectories Rendered in Java3D Using TrackerMonitor and TrackerDLL.

Presently, it is only possible to view the potentials at the end of the calculation.

4. DEVELOPER'S MANUAL

This section is taken directly from the *DynaPAC* manual.

4.1 DynaPAC Software Structure

4.1.1 DynaPAC Directory Tree

In this section we describe contents of the $DYNAPAC directory and its subdirectories. We use an appended slash (/) to indicate a directory; absence of a slash indicates a file.
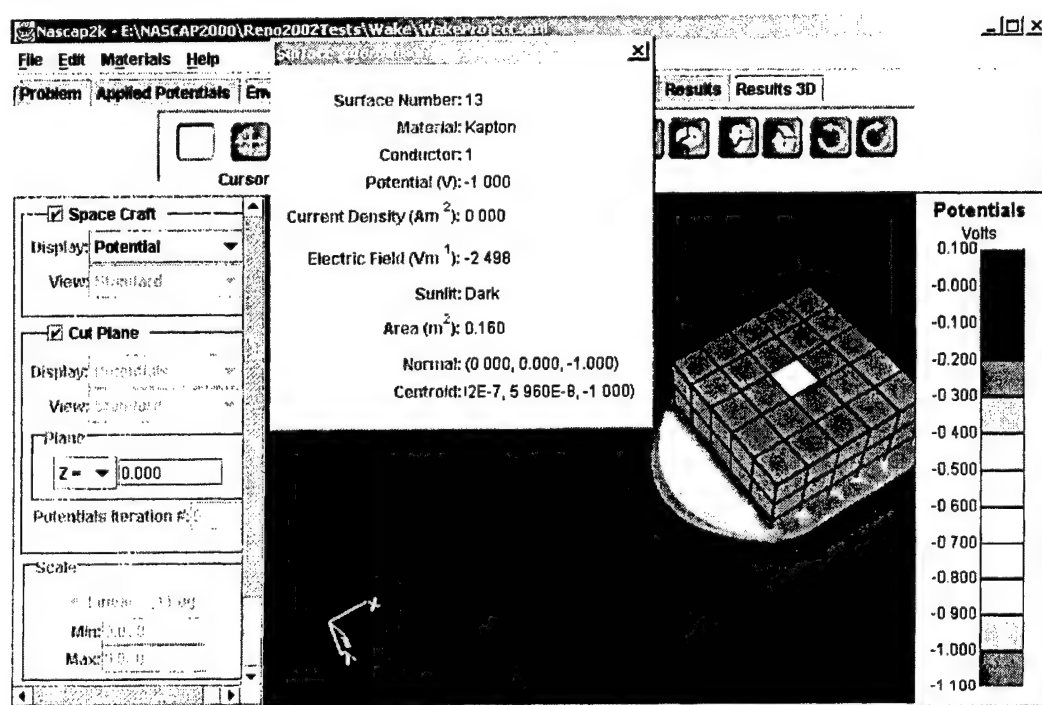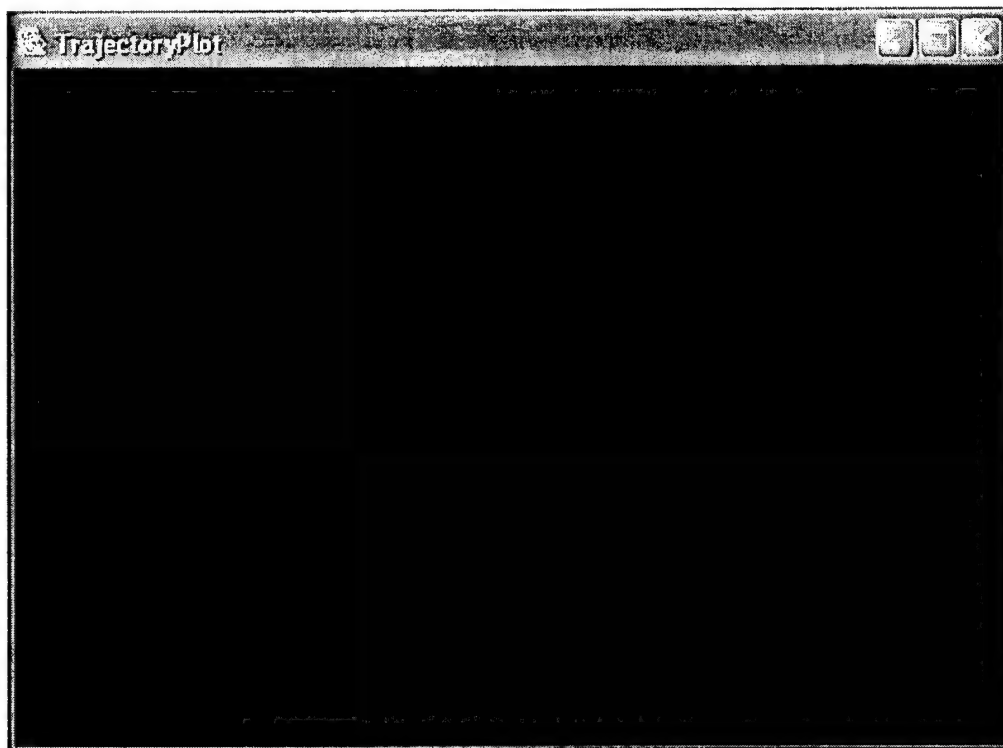
DynaPAC.setup -This is the file establishing environment variables and aliases for a *DynaPAC* user or developer.
Makefile - This should make all the *DynaPAC* modules.
bin/ - Directory for *DynaPAC* executable modules and shell scripts.
documentation/ -Contains this documentation
dynamake - Shell script which runs *DynaPAC* Makefiles.
inputs/ - Contains screen definitions and defaults for the *DynaPAC* interactive modules.
install - Shell script which installs *DynaPAC*.
lib/ - Contains *DynaPAC* libraries.
src/ - Contains the source code, object files, and include files for the *DynaPAC* modules and libraries.
testcases/ - Contains examples.
src/AddGrid/ - Code to add an outer grid to a *DynaPAC* grid structure.
src/DynaPost/ - Code for the DynaPost module.
src/DynaPre/ - Code for the DynaPre module.
src/GridTool/ - Code for the GridTool module.
src/MenDyn/ - Code for the MenDyn module.
src/NisDyn/ - Code for object definition interface to EMRC DISPLAY-II.
src/ObjPotl/ - Code for the ObjPotl module.
src/PartGen/ - Code for the PartGen module.
src/PatDyn/ - Code for the PatDyn module.
src/PolDyn/ - Code for the PolDyn module.
src/Potent/ - Code for the Potent module.
src/Scanner/ - Code for the Scanner module.
src/Tracker/ - Code for the Tracker module.
src/cadlib/ - Code for the library of object definition interface routines.
src/currents/ - Code for the library of routines used in the DynaPost/Currents module.
src/dblib/ - Code for the DataBase Manager library routines.
src/dynalib/ - Code for utility routines used by PartGen and Tracker.
src/ipslib/ - Code for routines used by DynaPre/IPS.
src/s3utils/ - Code for the low-level utility routines used by all *DynaPAC* modules.
src/*/Dbx/ - Source code resulting from processing by the 'cpp' preprocessor.

4.1.2 *DynaPAC Software Conventions*

There are many *DynaPAC* software conventions. Sometimes they are followed. Other times they are not.

Always use "syopen " to obtain a logical unit number and open a file. Get rid of it with "syclos."

4.1.3 *Modifying DynaPAC Subroutines*

Always type '../../dynamake' to remake a module.

In any of the $DYNAPAC/src/*modulename* directories you find source files with suffixes .f, .F, and .h.

If you modify a .f subroutine, you may compile it in the usual way, then type '../../dynamake' to remake the module. If you do not manually compile it, it should be compiled by the Makefile.

If you modify a .F subroutine, the Makefile should run the "CPP" processor to produce Dbx/name.f, compile Dbx/name.f, and remake the module.

If you alter a .h include file, the Makefile should reprocess all the necessary routines. Note that occasionally include files are shared between modules.

If you add a routine, be sure to add it to the relevant list in the Makefile.

If you alter a library routine, you may manually compile it and add it to the library, or you may type '../../dynamake' to take the library apart, compile the routine, and reconstruct the library.

4.2   Using the DataBase Manager

The following description of the DataBase Manager appeared in the Interim Report, SSS-DPR-90-11973, 30 September 1990. Check in the *DynaPAC* modules for further examples on the use of the DataBase Manager.

4.2.1 *DataBase Library*

The data base library provides the analysis and utilities functions necessary to define, manage, store, and retrieve data. In addition to the three callable routines (dbinfo(), dbdata(), and dbfile()) normally used to integrate the data base into an application, a stand-alone driver (dbtool) is also available to interact directly with the data definitions, data, and their files.

When a *DynaPAC* module is linked to the data base library, the following routines provide memory, data, and file management. The dbinfo() routine is used to define data base building blocks. Typical information commands define grids, grid nesting structures, data, lists, and

elements. The dbdata() routine is used to allocate/release reusable memory, read/write from disk files to memory, and initialize data values in memory. The dbfile() routine provides the means to manage the data files and their contents.

The input to these routines is in the form of strings containing keyword commands. The output is returned to the calling routine via common blocks within specialized include files. The calling routine saves the information it requires in its own data structures.

The data base can be thought of as a utility driven by its own keyword input language. Although some defaults are built into the data base, in most cases, the data required to perform a calculation must be constructed before it can be used. The dbinfo() routine is used to define data and the dbfile() routine is to define files.

The reader is referred to the data base header file, DBhead, for a complete keyword definition and current implementation status. Some examples of the use of the library routines are presented below. The first three demonstrate some data base functions. The final two examples show how a calculation using data in the data base memory storage area might be used.

4.2.2 Example: File Definition

The Task

A calculation wants to open a set of *DynaPAC* files that have the prefix 'Demo_Files' and then close them when it finishes.

The Solution

To open the files, use the command

    call dbfile('open prefix=Demo_Files')

This command assumes the default file type is DYNAPAC. A more explicit command would be

    call dbfile('open prefix=Demo_Files file_type=DYNAPAC')

To close just the Demo_Files file set, use

    call dbfile('close prefix=Demo_Files')

If desired, errors opening or closing the files can be checked by examining the error flags in the dbfile() output include file, 'odbfile.h.' The logical flag, 'ldferr,' is TRUE if an error was detected. The character string, 'cdferr,' contains an error message after error detection.

4.2.3 Example: Definition of List Dependent Data

The Task

Define a Surface_List list type dataitem, 'Surface_Centroids.'

The Solution

To define a data item that is a list, both the list type and the data item must be defined.

Two definitions are helpful since the list type definition depends on the particular object being defined, while this data item does not. Since the number of surfaces, or length of Surface_List is not given, let's define the data item in steps first.

Define the data type of Surface_Centroids.

```
call dbinfo('Define Data=Surface_Centroids Data_Type=LIST')
```

And it is which kind of LIST?

```
call dbinfo('Define Data=Surface_Centroids List_Type=Surface_List')
```

Each surface in the list has three real values, a location in three space or an offset from one of the corners. It depends on how the data is used by the application program. The data base does not know the difference.

```
call dbinfo('Define Data=Surface_Centroids Data_Dim=3 value_type=REAL')
```

All three of these commands could be combined into a single command. (Pretend all of the keywords are in one string and do not have embedded carriage returns.)

```
call dbinfo('Define Data=Surface_Centroids Data_Type=LIST
        List_Type= Surface_List Data_Dim=3 value_type=REAL')
```

Later when we find out how many surfaces there are on this object, the Surface_List can be defined. If there are 1,001 surfaces, then the data definition can be completed.

```
call dbinfo('Define List_Def=Surface_List List_Length=1001')
```


4.2.4 Example: Definition of Grid Dependent Data

The Task

Define a grid dependent data item, 'Ion_density,' which is element-centered.

The Solution

As in the example before, the data item definition is independent of the grid and element definitions. The data is defined as being of data type, 'SPATIAL.' This implies a different set of data is required for each grid in the problem. SPATIAL data is defined on the level of an element. Grids are thought to be rectangular lattices with equal spacing in each direction. An element is located at each lattice point. One of the predefined data items is one called 'ELEMENT_CENTERED.' We can save some work defining Ion_density by taking advantage of an existing definition using the 'SAME_AS keyword.'

        call dbinfo('DEFINE Data=Ion_density Same_As=ELEMENT_CENTERED')

The element type, ELEMENT_CENTER, is also predefined.

4.2.5   Sample Routine 1

This routine demonstrates how to use statement addressing functions to use the data in the data base manager.

```
        subroutine foo
        ...
        include "odbdata.h"
        include "odbinfo.h"
        include "data.h"
        ...
cc
cc *define local variables
        character*80 scomnd
        ...
cc
cc *define statement functions
     iaddrs(ix,iy,iz) = iofset+((iz-1)*ny+(iy-1))*nx+(ix-1)
cc
cc *find out about grids, save the number of grids
        call dbinfo("Inquire Problem")
        ngrids = idingd
cc
cc *loop on grids
        do 700 igrid = 1, ngrids
cc
cc *read in potentials for this grid
        write(scomnd,2000) igrid
 2000    format( 'READ DATA=pot_20_node grid=',i6)
        call dbdata( scomnd )
cc
cc *grab stuff out of output common (odbdata.h) which will be useful later
```

```
                iofset = idboff(1)
cc
cc *get information about the potentials in this grid
            write(scomnd,2500) igrid
  2500     format( 'INQUIRE DATA=pot_20_node grid=',i6)
        call dbinfo( scomnd )
            nxpot = idihir(1)
            nypot = idihir(2)
            nzpot = idihir(3)
cc
cc *loop on nodes of potentials
            do 600 iz=1,nzpot

              ...
              apot = rdata(iaddrs(ix,iy,iz))
              ...
  600     continue
cc
cc *all done with this grid, write it out and release its allocated memory
            write(scomnd,7000) igrid
  7000     format( 'WRITE CLOSE DATA=pot_20_node grid=',i6)
            call dbdata( scomnd )
  700    continue

         ...
         return
         end
```

4.2.6   Sample Routine 2

This routine demonstrates how to write a wrapper routine to hide the database completely from the calculation routine, foobar().

```
         subroutine foo

         ...
         include "odbdata.h"
         include "odbinfo.h"
         include "data.h"

         ...
cc
cc *define local variables
         character*80 scomnd

         ...
cc
cc *find out about grids, save the number of grids
         call dbinfo("Inquire Problem")
         ngrids = idingd
```

```
cc
cc *loop on grids
        do 700 igrid = 1, ngrids
cc
cc *read in potentials for this grid
          write(scomnd,2000) igrid
 2000     format( 'READ DATA=pot_20_node grid=',i6)
        call dbdata( scomnd )
cc
cc *grab stuff out of output common (odbdata.h) which will be useful later
          iofset = idboff(1)
cc
cc *get information about the potentials in this grid
          write(scomnd,2500) igrid
 2500     format( 'INQUIRE DATA=pot_20_node grid=',i6)
        call dbinfo( scomnd )
          nxpot = idihir(1)
          nypot = idihir(2)
          nzpot = idihir(3)
cc
cc *call the calculation submodule
          call foobar(rdata(iofset),nxpot,nypot,nzpot)
cc
cc *all done with this grid, write it out and release its allocated memory
          write(scomnd,7000) igrid
 7000     format( 'WRITE CLOSE DATA=pot_20_node grid=',i6)
          call dbdata( scomnd )
 700    continue
        ...
        return
        end
        subroutine foobar(pot20s,nx,ny,nz)
        real pot20s( nx, ny, nz )
        ...
        return
        end
```

4.2.7 DBHead File

This file (DBhead) contains the headers for calls to data base routines.
The present data base routines are:

dbfile(ccomnd) : File manipulation commands.
dbdata(ccomnd) : Data handling commands, movement from disk/memory.
dbinfo(ccomnd) : Access to data about data. Grid and data type info.

In general, data base commands are given in keyword oriented strings. The order of the keywords is not important. The keywords are not case sensitive. The command strings can be any length.

Valid names of files and data types start with a letter followed by any combination of letters, numbers, and the character "_." Case is important for file names, but data names are converted to uppercase. Names are stored in 80 character variables presently, but file names on some machines will be truncated to something shorter. Note that a four character suffix will be added to file names in order to identify the format of their contents.

Data item names may have multiple fields. The name for a piece of data will be the data type name, followed by the rest of the fields separated by one space. This constructed name will be truncated to 80 characters. If truncation of a name occurs, an error message is generated. When defining a data item with multiple names, just use the first name to set its properties. Then when making dbdata() calls or inquires, use the entire name.

For now, spatial items are 3 dimensional. When the keyword reader gets smarter, keywords using input lists will use () to enclose the list. For example, "EDGE_LENGTH=(len_x,len_y,len_z)."

Output is returned via an include file. It is unwise to rely on the order of the contents of the include files since they may change. The names of the variables in the file should be constant.

It should be noted that the current implementation of time names has been done in such a way that it may be a useful guide for future modifications. Time stamps are simply names that can be used to group otherwise identically identified data items. If another or additional distinguishing names were desired, the time stamp/name usage could be generalized to allow a number of different and/or parallel sets of data grouping. The relationships of the groupings would be determined by a set of submodules or functional relationships.

BUGS: All errors are fatal right now.
    Not all commands are fully implemented.
        Defaults are not always documented.

The following notes are subject to change as development continues, but
        should be fairly stable.

The keywords have been marked to indicate implementation status.
        No mark means it is implemented and may even work.
        % - Partially implemented. Restrictions apply.
        + - Partially implemented. Recognized, but ignored.
        * - Not implemented. Will generate fatal error.

Keyword defaults are in [ ]'s, where appropriate.

The predefined items are described at the end of the file.

Some example routines using the data base calls are available in the
dynapac directory on s3dawn in ~dynapac/Notes/Examples.

Files Oriented
dbfile(ccomnd) - interface routine for file handling
ccomnd is a keyword oriented input string like:
"Open Prefix=Dmsp"
"STATUS prefix=Data_set_1 PREFIX=Data_set_2"
output from the routine is returned in "odbfile.h"

command keywords are :
EXIT - Close all open files.
OPEN - Assign file prefix(s) to run.
CLOSE - Close file prefix(s).
* STATUS - Query current file status of prefix(s).
* WHAT - STATUS of all known prefixes.
* HELP - This list.
identifying information:
PREFIX=File_Name - Define a file prefix. [last prefix used]
FILE_TYPE=keyword - Specify a file type. (DYNAPAC)
SAVE_FILE=logical - save file when done[TRUE] or delete(FALSE)
+ DIAGNOSTIC=keyword - turn ON,[OFF] diagnostics

Data Oriented
dbdata(ccomnd) - interface routine for data base requests
ccomnd is a keyword oriented input string like
"read data=pot_20_node grid=2"
output from the routine is returned in "odbdata.h"
command keywords are broken into groups below:
 action instructions:
OPEN - Allocate space for data in memory.
CLOSE - Deallocate space for data in memory.
* CRUNCH - Squeeze the gaps out of memory.
% READ - Transfer data from disk to memory.
(% - must open disk prefix first)
% WRITE - Transfer data from memory to disk.
(% - must open disk prefix first)
* COPY - Copy first data name to rest of data names (in memory).

%   INITIALIZE - Initialize a data type to its "set_value."
                (% - only dimension=1 items)
+   DELETE - Remove a data type from disk.
                (% - empty space in file is not reclaimed)
         OVER_WRITE - Replace data on disk defined by OLD_DATA, OLD_GRID,
                and/or OLD_TIME with the data defined by DATA, GRID,
                and/or TIME. Will use names of new data (DATA, GRID,
                and TIME) as defaults for undefined old data
                variables (OLD_DATA, OLD_GRID, and OLD_TIME).
         ADD_GRID   - Add outer grid to existing grid structure.
                All grid-dependent data will be renamed to reflect the
                change in grid numbering. New primary grid will carry
                parameters of old primary grid.
+   DIAGNOSTIC=keyword - turn ON,[OFF] diagnostics
*   HELP - This list.
  identifying information:
         DATA=name - Specify a data name (repeatable).
         GRID=name - Specify a grid number or "ALL" grids.
         TIME=name - Specify the time stamp.
         PREFIX=file_name - Specify a file prefix.
         OLD_DATA=name - Specify old data name. (See OVER_WRITE)
         OLD_GRID=name - Specify old grid number. (See OVER_WRITE)
         OLD_TIME=name - Specify the old time stamp. (See OVER_WRITE)
  optional commands to override data base default actions:
         LENGTH=int - Size in words needed for data in memory. Used
                  to modify data type definition temporarily. Normally
                  this keyword is only used to OPEN data items with
                  multiple field names. In this case, the data type
                  is defined with by the first field only while locations
                  in memory are associated with the full name.
         HERE=int - Point to a memory address, use idbloc() to get
                  offset from idata(0). Used to tell the data base where
                  something is. Note that much of the internal error
                  checking will be defeated by this command.

Grid and Data Definition/Type Oriented
         dbinfo(ccomnd) - interface routine for defining and learning
                  information about grid definitions and data typing
                  properties. Sorry, only one PROBLEM, GRID, ELEMENT, DATA,
                  or INTERP per call. Use dbdata("STATUS DATA=stuff GRID=2")
                  to find the memory offset for the "stuff" data in grid 2.

                  a dbinfo() command is a keyword oriented input string like:

63

```
"DEFINE Data=pot_8_node element_type=Node_8
        dimension=scalar value=real set_value=0. "
```

output from the routine is returned in "odbinfo.h"

command keywords are :
        DEFINE - Define or redefine a piece(s) of information.
        INQUIRE - Request information about a GRID, ELEMENT, DATA, or
                LIST_DEF, INTERP. (see context keywords)
    +   DIAGNOSTIC=keyword - turn ON,[OFF] diagnostics
    *   HELP - This list.

context keywords:
        PROBLEM - Get the generally useful stuff about problem.
        DATA=name - Commands are in regards to data type "name."
                if Command = DEFINE then only first name field is needed
                if Command = INQUIRE then use all name fields to get
                        specific information. Otherwise, just get
                        general information.
        ELEMENT=name - Commands are in regards to element type "name."
        LIST_DEF=name - Commands are in regards to list type "name."
        GRID=name - Commands are in regards grid type "name."
                Only one grid's worth of information.
    *   INTERP=name - not implemented. "linear,""quadratic."
        SAME_AS=name - The new thing is the "same as" name.

DATA parameters:
        DATA_TYPE=keyword - this data is SPATIAL, BUFFER, LIST, etc
        DATA_LENGTH=integer - number of elements in a BUFFER
        ELEMENT_TYPE=element_name - flag describing what this value is
        LIST_TYPE=list_name - name of list definition for this data type
        DATA_DIM=dim_flag - either a number (values/pt.) or flag
        VALUE_TYPE=value_type - type of values at each point
                value_type = <REAL/INTEGER/LOGICAL/OCTAL/ALPHA>
        ALPHA_LENGTH=nchars - length of ALPHA types in characters [80]
    %   SET_VALUE=value - default initialization value/method.
                (% - only know how to set DATA_DIM=1 to constants)
        SAVE_DATA=logical - save data on perm.[TRUE] or temp.(FALSE)
        OWN_FILE=keyword - [FALSE] - put this data in the main file
                        - TRUE - same as DYNAPAC below
                        - DYNAPAC - put data in own standard subfile
                        - LONG_NUMBER <number_subkeys> - for data
                          types with many subkeys (>~200).
                          <number_subkeys> is maximum number of
                          keys to put in the file (default is
                          1000). Addressable only with

64

DATA_NAME <number>, where <number> is
between 1 and <number_subkeys>.
FILE_SUFFIX=name - name to use for OWN_FILE suffix (should
include .""" as first character)
GRID_DEPEND=logical - TRUE if data is grid dependent. [FALSE]
TIME_DEPEND=logical - TRUE if data is time dependent. [FALSE]
+ TIME_SAVE=integer - how many time steps of data to save [1]
Automatically OVER_WRITEs oldest (smallest time stamp)
version of data.

ELEMENT parameters:
UNIT_SIZE=integer - how many sets of values are in each unit
UNIT_OFFSET=offset_x,offset_y,offset_z - offset in mesh units
from element origin, should be of approp.
dimension and there should be one offset for
each "UNIT_SIZE" for SPATIAL items.
+ INTERP_TYPE=name - method of interpolation between points

LIST_DEF parameters:
LIST_LENGTH=integer - number of entries in the list_name

INTERP parameters: none for now, just the name. Will recognize
"linear," "quadratic."
maybe later will add ax**3+bx**2... type stuff.

GRID parameters (note GRID is a list of SPATIAL elements):
OUTSIDE - this grid encloses the other grids(if any)
INSIDE=grid_name - this grid is inside grid_name
GRID_SIZE=real - grid mesh size (meters), redefines all grids!
MESH_RATIO=integer - ratio of inner/outer mesh units (>=1)
PRIM_RATIO=integer - ratio of inner/primary (or outer most)
mesh units
ORIGIN=value_x,value_y,value_z - In the INSIDE grid the
origin is in outergrid units. For OUTSIDE
grids, the origin is the location of the old
grid 1 in this grid.
(always true: ORIGIN=1,1,1 in outermost grid)
ORIGIN_PRIM=value_x,value_y,value_z - Origin location in
primary or outer most grid units.
EDGE_LENGTH=length_x,length_y,length_z - number of nodes
along the grid edge, includes endpoints.
% GRID_INTERP=name - outer grid interpolation function
(% - =0 for now)

65

The following items are predefined :
(<none> means no default value is set for that attribute)

Data Types [OwnFile=FALSE]
  BUFFER Data types

Name	DatTyp	ValType	Len	SaveDat	InitVal	Dim	GrdDep	TimDep
BUFFER_DATA	BUFFER	INTEGER	0	FALSE	' 0'	1	FALSE	FALSE

  LIST Data types

Name	DatTyp	ListTyp	ValType	SaveDat	InitVal	Dim	GrdDep	TimD
LIST_DATA	LIST	<none>	INTEGER	TRUE	' 0'	1	FALSE	FALSE

  SPATIAL Data types [ValType=REAL, InitVal=' 0.0', SaveDat=TRUE, DataDim=1,
              GridDep=TRUE, TimeDep=FALSE, NumTimSav=1]

Data Type Name	Element Type Name
SPATIAL_DATA	<none>
NODE_8	NODE_8_ELEMENT
NODE_20	NODE_20_ELEMENT
NODE_32	NODE_32_ELEMENT
ELEMENT_CENTERED	ELEMENT_CENTER

Element Types

Name	Pts/Elm	Interp	Pt Locs
ELEMENT_CENTER	1	LINEAR	(0.5, 0.5, 0.5)
NODE_8_ELEMENT	1	LINEAR	(0.0, 0.0, 0.0)
NODE_20_ELEMENT	4	QUADRATIC	(0.0, 0.0, 0.0)
			(0.5, 0.0, 0.0)
			(0.0, 0.5, 0.0)
			(0.0, 0.0, 0.5)
NODE_32_ELEMENT	4	QUADRATIC	(0.0, 0.0, 0.0)
			(0.5, 0.0, 0.0)
			(0.0, 0.5, 0.0)
			(0.0, 0.0, 0.5)

List Types (Lengths should be redefined)

Name	Length
SURFACE_LIST	0
SURFACE_NODES	0

Grids (the outermost grid should be redefined)

Name	MeshLen	Origin	EdgeLen	Grid_Interp
'1'	1. m	(1., 1., 1.)	(7, 9, 11)	NONE

---

end of file "DBhead"

66

5. EXAMPLES

5.1 Charging in a Tenuous Plasma

A simple spacecraft, illustrated in Figure 38, is created to illustrate the charging calculations. The direction of the solar arrays is appropriate to a spacecraft at 6 am local time.
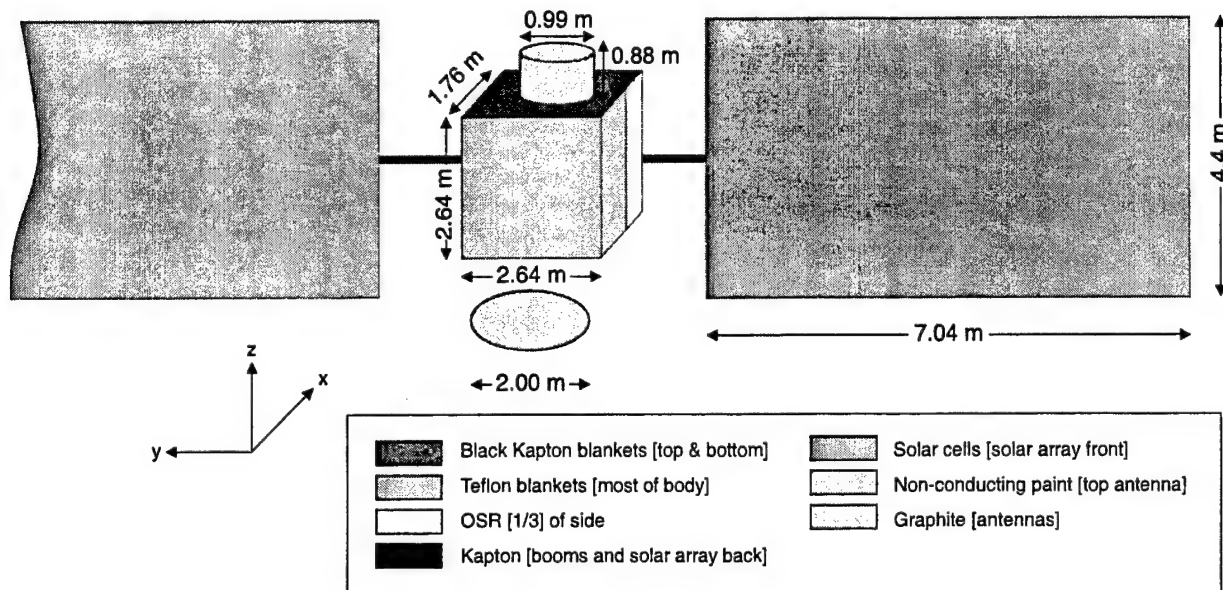
Figure 38. Illustrative Spacecraft Used for Sample Spacecraft Charging Calculations Using *Nascap-2k*.

The calculations use the "Worst Case" environment recommended by Reference 9 (see Figure 39) for initial modeling during the spacecraft design process. The spacecraft charges for 15 minutes, as this is longer than any spacecraft would be exposed to such a severe environment.

The sun is taken to be incident on the spacecraft from the (0.92, 0.39, -0.02) direction. This is appropriate to a spacecraft in geosynchronous orbit at 0 longitude at 6 am GMT on January 1, 2000, consistent with the geometry model.

67
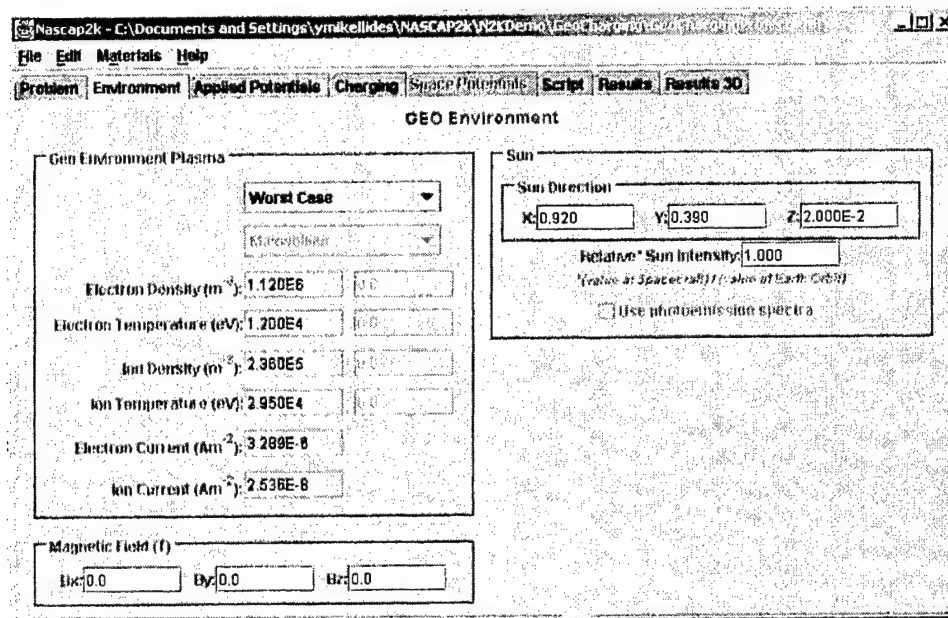
Figure 39. 90 Percent Worst-case Environment for Geosynchronous Orbits Used for All
Charging Runs as Defined in Reference 9.

5.1.1 Nascap-2k *Geometric Model*

Nascap-2k has a flexible geometric modeling capability. The user can control the size, shape, and gridding of each component. Figure 40 shows the model in Object Toolkit. The omni antenna is represented by an octagonal cylinder and the side antennas are concave dishes.
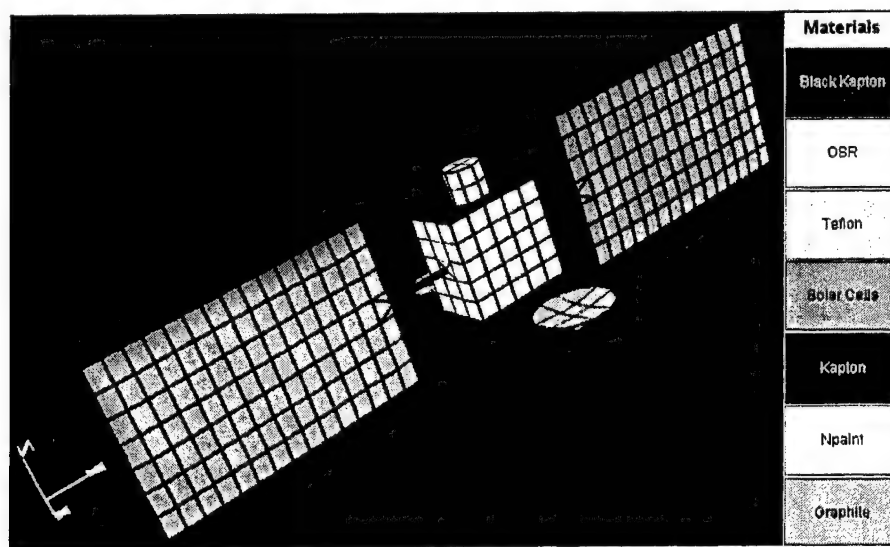
Figure 40. *Nascap-2k* Geometric Model of Spacecraft.

5.1.2   Results

The calculation was set up for 99 timesteps for a time period of 1000 seconds. The timesteps are geometrically distributed with a minimum of 0.1 seconds and a maximum of 60 seconds.

The results of this sample calculation are summarized in Table 10. The potentials at 1000 seconds are shown in Figure 41. The least negative surfaces are the ends of the solar arrays. The shaded Teflon surfaces are the most negative. All the surfaces that are more negative than the chassis are shaded insulators. In the center of the sun-facing side of the spacecraft body, the Teflon is slightly positive with respect to the chassis. The sunlit insulators on the body are near the chassis potential or positive with respect to the chassis.

Table 10. Results of Calculations (potential in kV).

	Chassis	Kapton	OSR	Solar Cells	Teflon	Non-conducting paint
Absolute potentials	-12.0	-11.5 to -14.4	-10.0 to -13.7	-7.2 to -10.8	-7.9 to -14.0	-10.0 to -12.2
Differential potentials		0.5 to -2.4	2 to -1.7	4.8 to 1.2	4.1 to -2	2 to -0.2
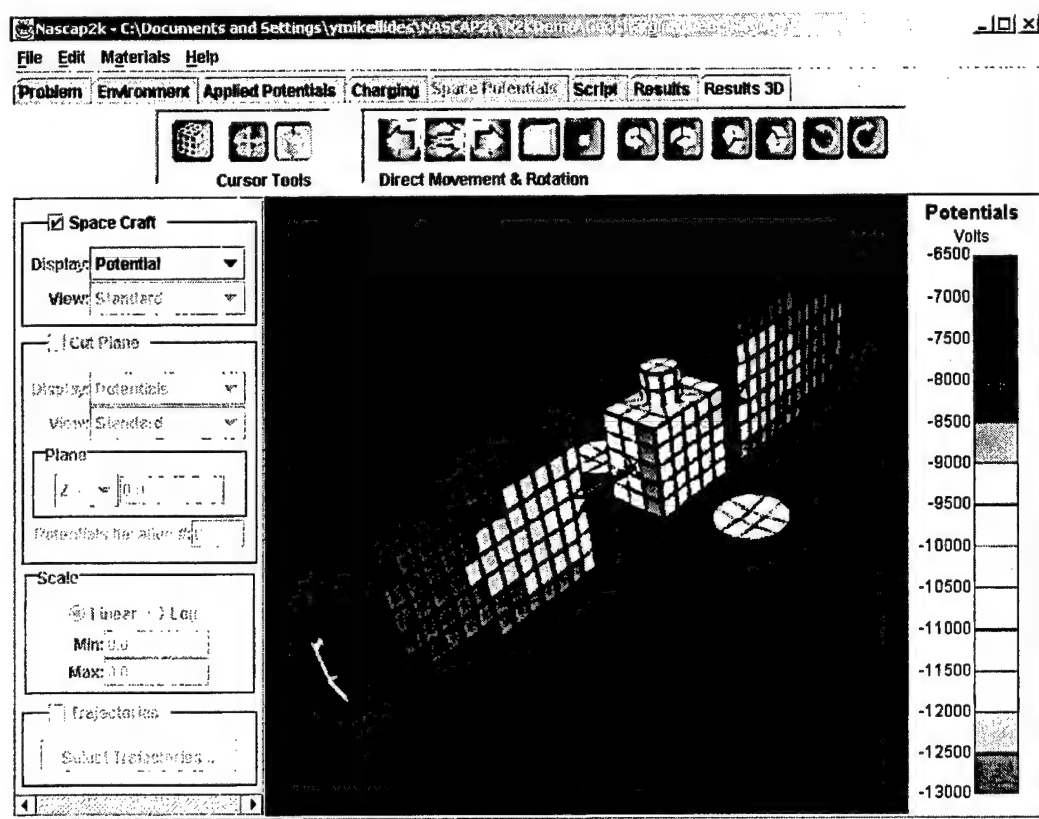
Figure 41. Results of Spacecraft Charging Calculation (at 1000 sec) Using *Nascap-2k*.

69
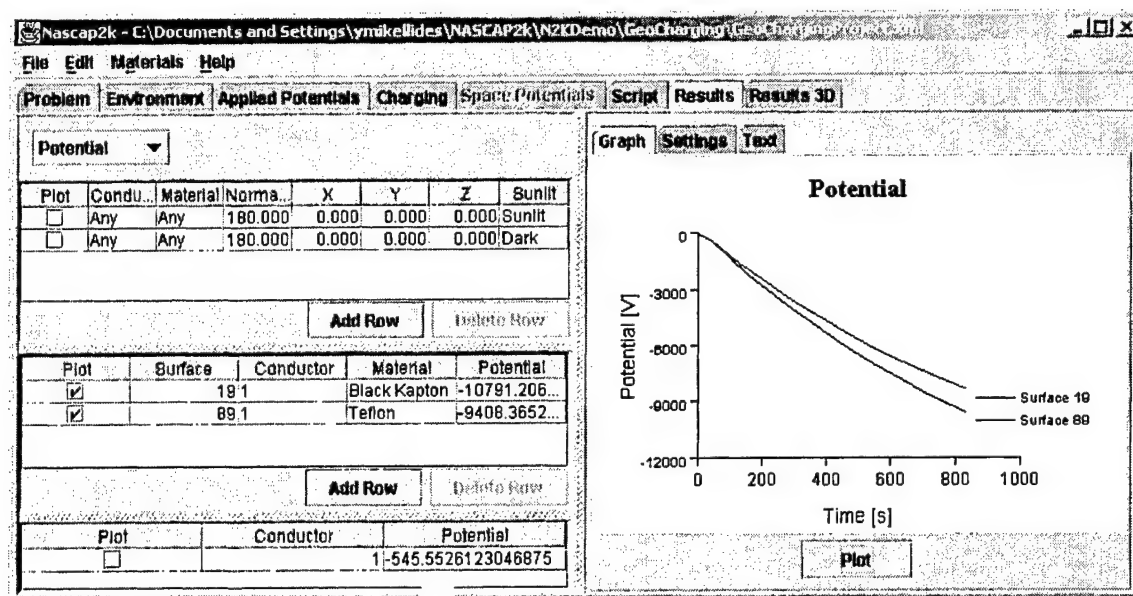
Figure 42. Potential as a Function of Time for the Black Kapton and Teflon Surfaces.

In order to obtain a stable solution, the variation of the potential within a single timestep is limited. *Nascap-2k* uses much less stabilization than other codes such as NASCAP/GEO and the *SEE Spacecraft Charging Handbook* because it uses more stable algorithms. The user is assumed to understand the code well enough to make the appropriate adjustments in the number and distribution of the timesteps in order to obtain a stable and accurate solution.

5.2   Single Cube

In this example *Nascap-2k* is used to compute the potential about a charged cube. The calculation proceeds as follows:

1) Use Object ToolKit to define an Aluminum cube with a side of 20 cm.

2) Use GridTool to define a grid about the cube with one level of subdivision.

3) Specify the environment

4) Initialize the potential of the cube at −1000 volts.

5) Calculate potentials in space

6) Display results

The first step is to create the object. To do that start the *Nascap-2k* GUI and in the opening screen choose "Create New Project." We choose the name of the project as "OneCube" which we save in a folder called "OneCube." This action generates an .xml file called "OneCubeProject.xml" and copies all the necessary schemas (Table 11) into the project's folder.

70

Table 11. Schema Fles.

File
Assembly_Schema.xml
BEM_schema.xml
Brick_Schema.xml
Cylinder_Schema.xml
Dish_Schema.xml
material_schema.xml
Mesh_Schema.xml

5.2.1 *Define the Object*

The sequence below takes the user through the steps of creating the cube.

1) In the "Problem" tab choose "Edit Object" which brings up the Object Toolkit window.

2) Under "Component" choose "Add New" → "Box."

3) In the object specification window choose 4 zones for each direction, and a length of 0.2 m for all sides of the cube. Define the material of all surfaces to be aluminum; this is also the only conductor, numbered 1 (Figure 43, bottom).

4) Save the object. We choose the name "OneCube" creating the corresponding .xml file "OneCube.xml."

5) Exit Object ToolKit

5.2.2 *Define the Grid*

The sequence below takes the user through the steps of creating the grid. We choose a two-level grid as described below:

1) In the "Problem" tab choose "Edit Grid" which brings up the GridTool window.

2) Choose "File" → "Import Object" → "OneCube.xml" to import the object into GridTool. The cube now appears in the GridTool display.

3) Choose "Grid" → "New Primary Grid" and specify a 12x12x12 grid (in the "Grid Dimensions" fields). Also choose a 0.6 m mesh size (Figure 44, top).

4) Choose the newly created primary grid (by clicking on the folder icon labeled "1") and then choose "Grid" → "New Child Grid" In the "Child Grid" screen, specify the grid limits as 5 and 9 for all three directions. For the subdivision ratio choose 4 (Figure 44, bottom).

5) Save the grid and then exit GridTool. This action has created the *prefix.grd* file (see Table 3 for the interpretation of the contents).

71

5.2.3   Perform the Calculation and Display Results

The problem is now set for the initialization and calculation of physical parameters. Once back in the main "Problem" tab choose "File" → "Load Object." This action enables the environment options and type of calculation.

For this example we choose a LEO orbit in the interest of computing potentials in space. In the "Environment" tab we specify a density of 10^{11} m^{-3}. We retain all the remaining default values (Figure 45, top).

In the "Applied Potentials" tab, set the conductor boundary condition at "Fixed Potential" with a value of –1000 V (Figure 45, middle).

In the "Space Potentials" tab choose the "Laplace" charge density model (see section 2.9.1), and under "Advanced Potential Solver Parameters" change only "Maxits," "RDRMin" and "Maxitc" from their default values to 10, 1e-3 and 40, respectively.

In the "Script" tab choose "Build Script" and then "Run Script" to perform the calculation. The following screens appear to monitor the evolution of the calculation: "BEM Monitor," "N2kDyn Monitor" and "Potent Monitor."

In the "Results 3D" tab, the object appears at the specified (fixed) potential (-1000 V). To view potentials in space select an appropriate "CutPlane" (Figure 47).
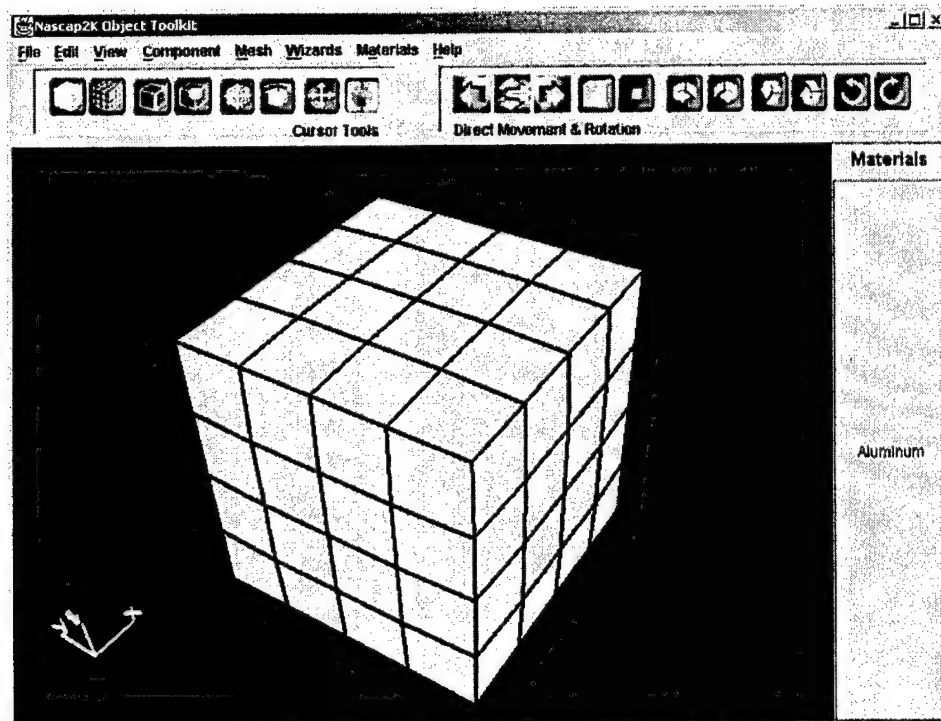
Figure 43. Object Definition for the "OneCube" Sample Problem.
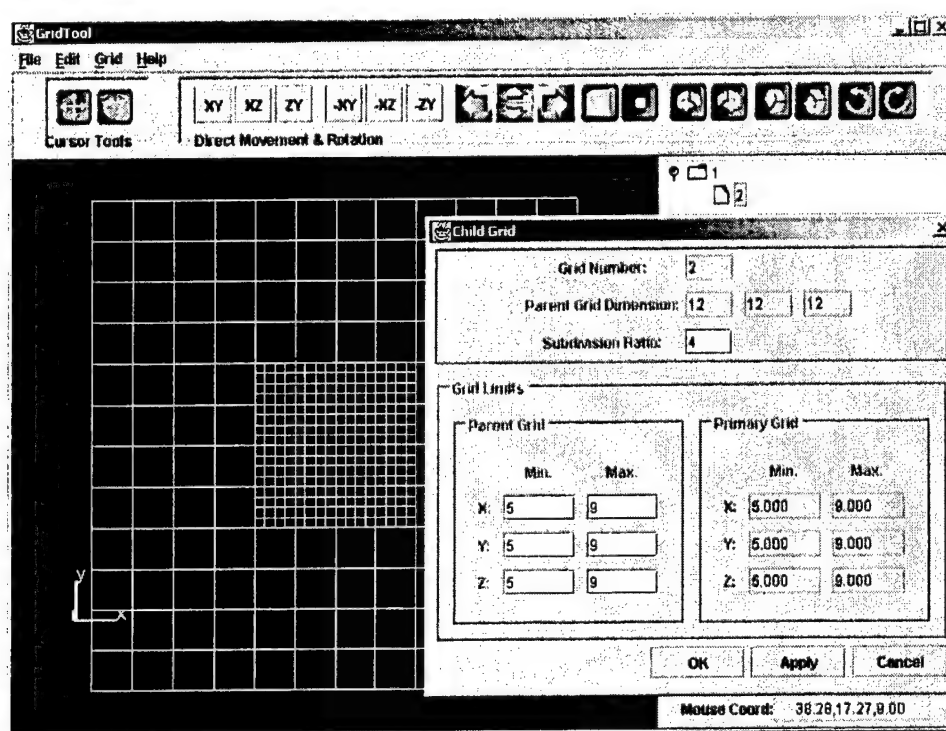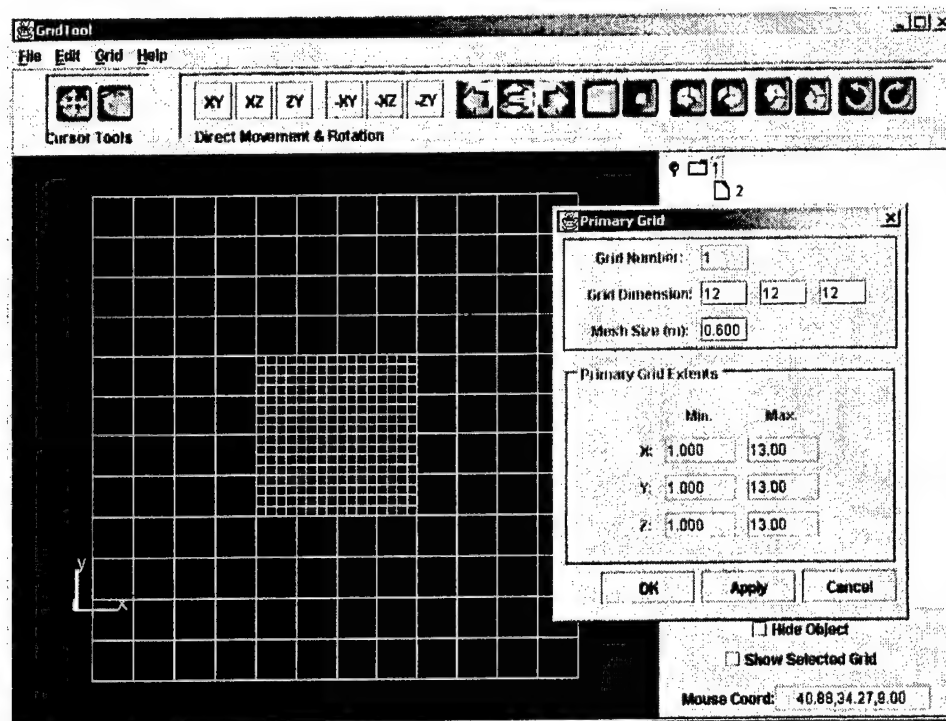
Figure 44. Grid Definition for the "OneCube" Sample Problem. Top: primary grid definition. Bottom: child grid definition.

Figure 45. Computing Potentials in Space for the "OneCube" Sample Problem. Top: specify the environment. Middle: define applied potentials. Bottom: specify parameters for the potential solver.
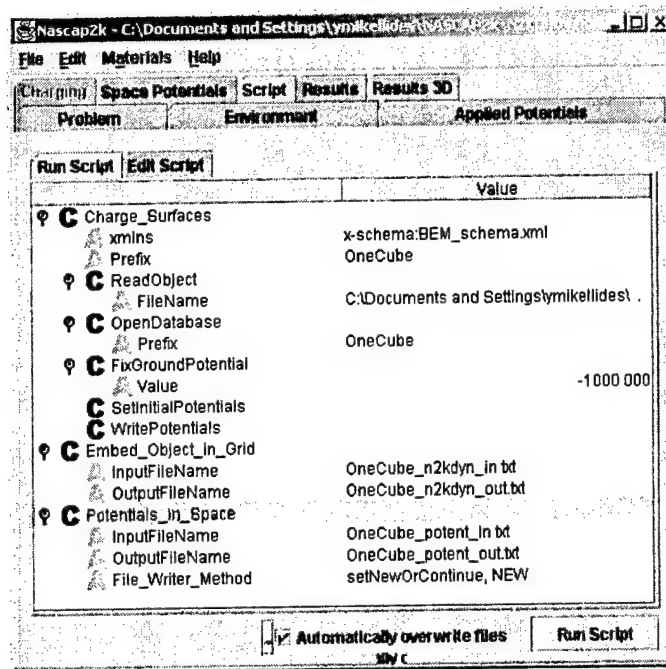
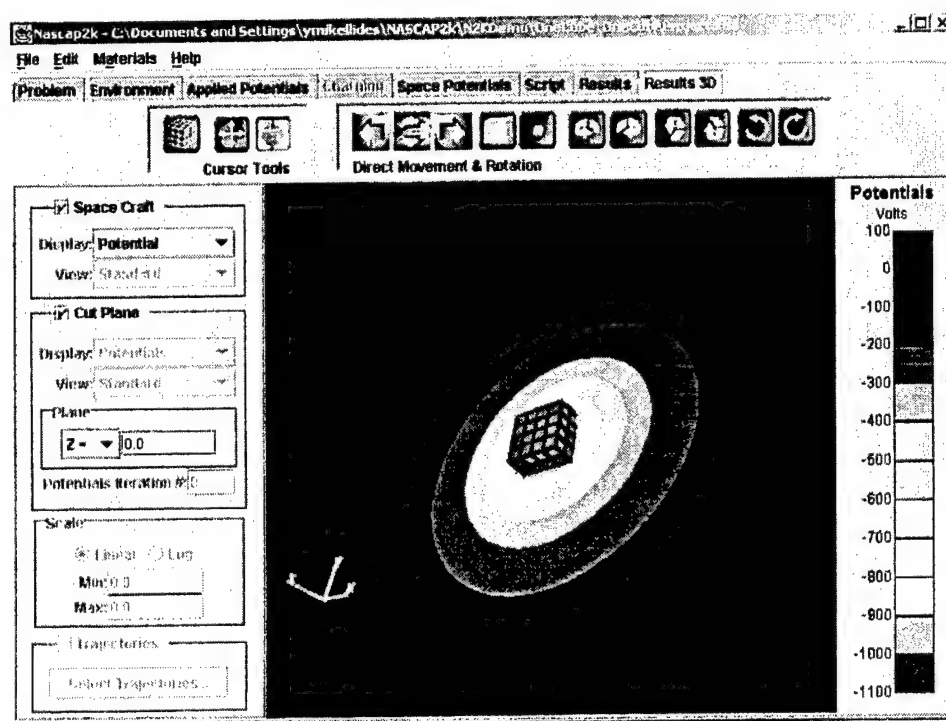Figure 46. Script for the "OneCube" Sample Problem.

Figure 47. Computed Potential Contours in the X-Y Plane (Z Normal) for the "OneCube" Sample Problem.

76

5.3 Two Cubes

In this example *Nascap-2k* is used to compute the potentials in space about an object consisting of two disjoint, unequal size cubes. The calculation proceeds similarly to the previous example.

5.3.1 Define the Object

The sequence below takes the user through the steps of creating the two cubes.

In the "Problem" tab choose "Edit Object" which brings up the Object ToolKit window. Under "Component" choose "Add New" → "Box."

In the object specification window choose 4 zones for each direction, and a length of 0.2 m for all sides of the cube. Define the material of all surfaces to be aluminum; this is also conductor number 1. From the cursor tools choose the "Node Tool" and using the mouse select the center node on the surface with a normal parallel to the Z-axis. Then choose "Component" → "Node Relative Move" and the X-node position to –0.2 m (Figure 48).
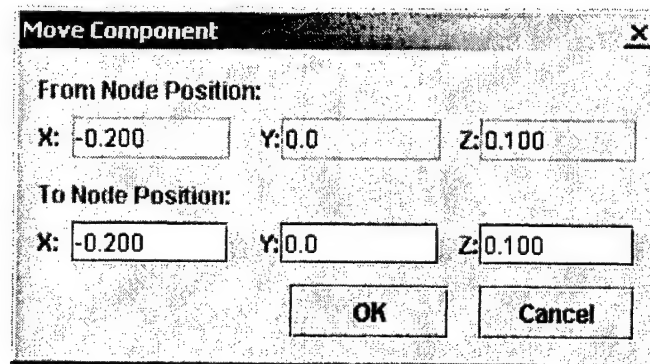

Figure 48. Object ToolKit Screen for Defining Node Positions.

1) Repeat step 2 to create the second cube. Choose 4 zones and a size of 0.3 m. Select Gold as the material; this is conductor 2.

2) Set the X-position of the center node on the surface perpendicular to the Z-axis at +0.25 m. The surfaces of the object that face each other are now 20cm apart. Choose "Components" → "Consolidate Components into Assembly" to combine the two cubes into one object.

3) Save the object as "TwoCubes" and exit Object ToolKit.

5.3.2 Define the Grid

The sequence below takes the user through the steps of creating the grid. We choose a two-level grid as described below:

1) In the "Problem" tab choose "Edit Grid" which brings up the GridTool window.

2) Choose "File" → "Import Object" → "TwoCubes.xml" to import the object into GridTool. The cubes now appear in the GridTool display.

3) Choose "Grid" → "New Primary Grid" and specify a 12x12x12 grid (in the "Grid Dimensions" fields). Also choose a 0.16 m mesh size.

4) Choose the newly created primary grid (by clicking on the folder icon labeled "1") and then choose "Grid" → "New Child Grid" In the "Child Grid" screen, specify the grid limits as 4 and 11 for the X-direction, and 5 and 9 for Y,Z-directions. For the subdivision ratio choose 4.

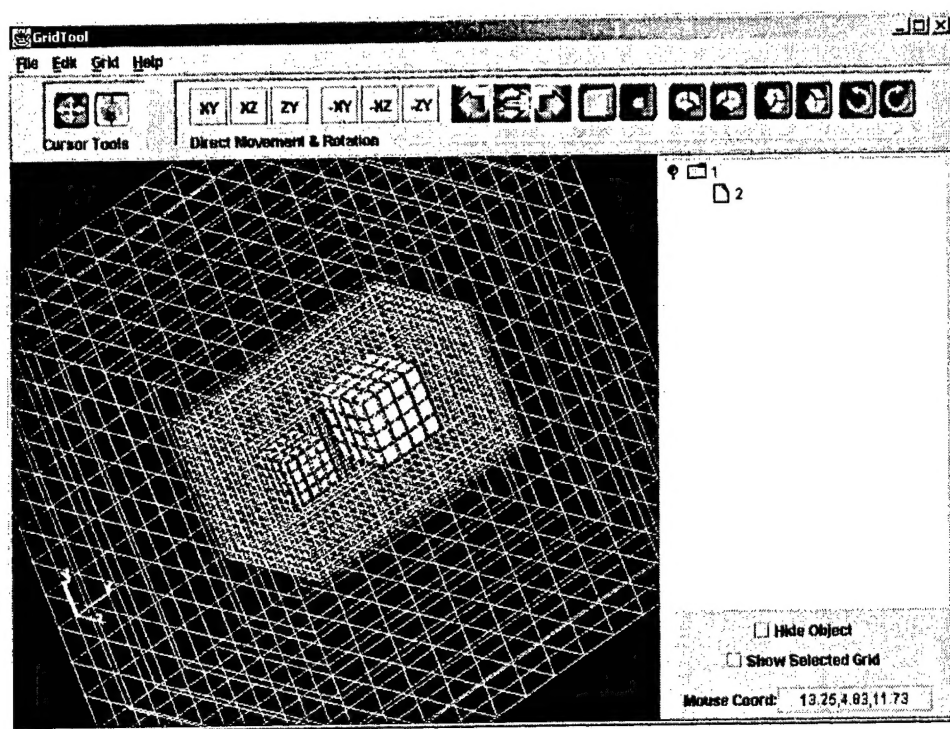5) Save the grid and then exit GridTool. The grid just created is shown in Figure 49.

Figure 49. Grid for the "TwoCubes" Sample Problem.

5.3.3 Perform the Calculation and Display Results

The problem is now set for the initialization and calculation of physical parameters. Once back in the main "Problem" tab choose "File" → "Load Object." This action enables the environment options and type of calculation.

1) For this example we choose a LEO orbit in the interest of computing potentials in space. In the "Environment" tab we specify:

   Density = 1e11 m^{-3}
   Temperature = 0.1 eV
   Species: O+ and e-
   Bx = 0, By = 0, Bz = 4e-5 T.
   We retain all the remaining default values.

2) In the "Applied Potentials" tab, set the conductor 1 boundary condition at "Fixed Potential" with a value of −10 V (this is the small cube). Set the conductor 2 boundary condition at "Biased Potential from Conductor #1" with a net value of +10 V as shown in (this is the large cube). This action biases the large cube's potential at +10 relative to the small cube's.
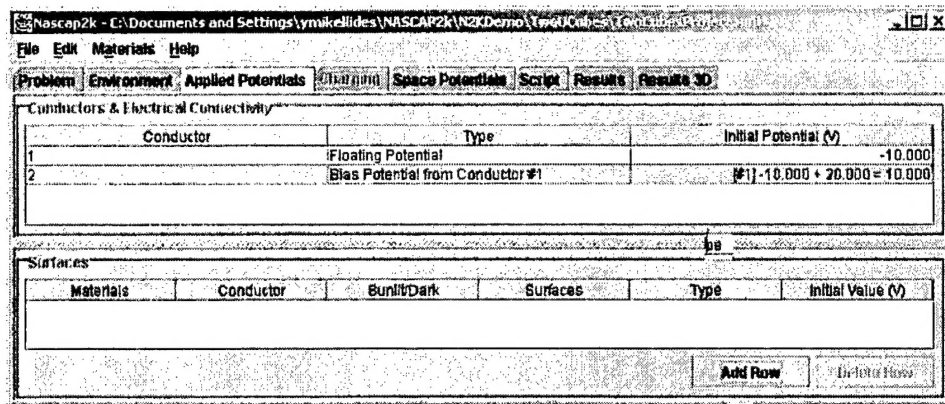

Figure 50. Initializing Potentials for the "TwoCubes" Problem.

3) In the "Space Potentials" tab choose the "Non-Linear" charge density model (see section 0), and under "Advanced Potential Solver Parameters" change only "Maxits," "RDRMin" and "Maxitc" from their default values to 10, 1e-3 and 40, respectively (see section 5.3).

4) In the "Script" tab choose "Build Script" and then "Run Script" to perform the calculation. The following screens appear to monitor the evolution of the calculation: "BEM Monitor," "N2kDyn Monitor" and "Potent Monitor." The action creates the files listed in Table 2 (in the project folder).

5) Once at the "Results 3D" tab, the object appears as in Figure 51.
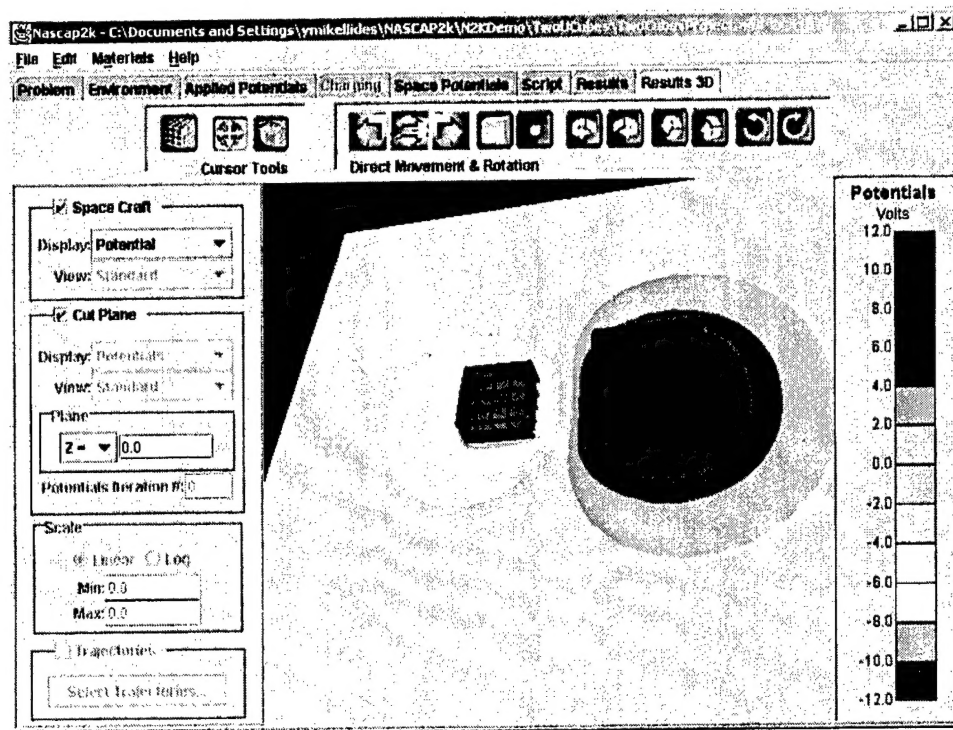
Figure 51. Potential Distribution for the "TwoCubes" Sample Problem.

REFERENCES

1   M.J.Mandell, I. Katz, J.M. Hilton, J. Minor, D.L. Cooke, (2001) *Nascap-2k*, A Spacecraft Charging Analyis Code for the 21st Century, AIAA Paper 2001-0957, *AIAA Aerospace Sciences Meeting & Exhibit, 39th, Reno, NV.*

2   M. J. Mandell, P. R. Stannard, I. Katz, (1993) *NASCAP Programmer's Reference Manual,* NASA CR 191044.

3   M.J. Mandell, and I. Katz, (1990) High Voltage Plasma Interactions Calculations Using NASCAP/LEO, AIAA Paper AIAA-90-0725.

4   I. Katz, G.A. Jongeward, V.A. Davis, M.J. Mandell, R.A. Kuharski, J.R. Lilley, Jr., W.J. Raitt, D.L. Cooke, R.B. Torbert, G. Larson, and D. Rau, (1989) Structure of the Bipolar Plasma Sheath Generated by SPEAR I, *J. Geophys. Res., 94*, A2, p. 1450.

5   J.R. Lilley, Jr., D.L. Cooke, G.A. Jongeward, I. Katz, (1989) *POLAR User's Manual*, GL-TR-89-0307, ADA232103.

6   M.J. Mandell, T. Luu, J. Lilley, G. Jongeward, and I. Katz, (1992) *Analysis of Dynamical Plasma Interactions with High Voltage Spacecraft,* (2 volumes), Rep. PL-TR-92-2248, Phillips Lab., Hanscom Air Force Base, MA, ADA262784, ADA262783.

7   V.A. Davis, M.J. Mandell, D.L. Cooke, C.L. Enloe, (1999) High-voltage interactions in plasma wakes: Simulation and flight measurements from the Charge Hazards and Wake Studies (CHAWS) experiment, *J. Geophys. Res., 104*, A6, p. 12445.

8   M.J. Mandell, G.A. Jongeward, D.L. Cooke, W.J. Raitt, (1998) SPEAR 3 flight analysis: Grounding by neutral gas release and magnetic field effects on current distribution, *J. Geophys. Res., 101*, A1, p. 439.

9   C. K. Purvis, H. B. Garrett, A. C. Whittlesey, N. J. Stevens, (1984) *Design Guidelines for Assessing and Controlling Spacecraft Charging Effects*, NASA TP 2361.

10 V.A. Davis, L.W. Duncan, (1992) Spacecraft Charging Handbook, PL-TR-92-2232, ADA262788.

11 H.B. Garrett, (1981) The Charging of Spacecraft Surfaces, *Reviews of Geophysics and Space Physics, 19*, 4, p. 577.

12 E.C. Whipple, (1981) Potentials of Surfaces in Space, *Reports on Progress in Physics, 44*, p. 1197.

13 S. A. Brebbia, (1981) *Boundary Element Methods*, Springer Verlag, New York.

14 S. M. Selby, (1975) *Standard Mathematical Tables*, CRC Press, Cleveland, OH.

15 M. J. Mandell, I. Katz, G. W. Schnuelle, P. G. Steen, J. C. Roche, (1978) The Decrease in Effective Photocurrents due to Saddle Points in Electrostatic Potentials near Differentially Charged Spacecraft, *IEEE Trans. Nucl. Sci. NS-25.*